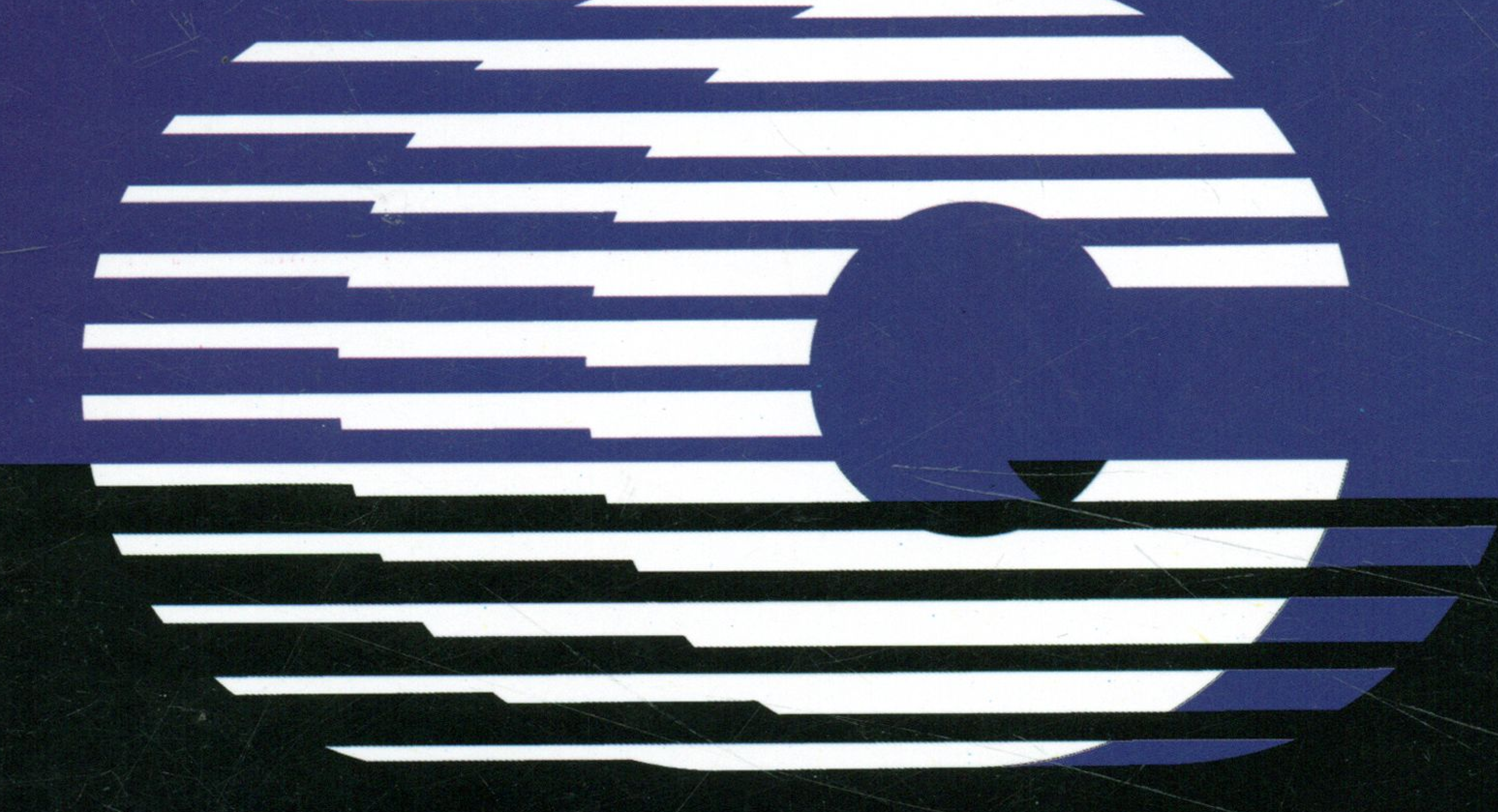
مقدمة إلى البرمجة بلغة سي



بشير علي القائد

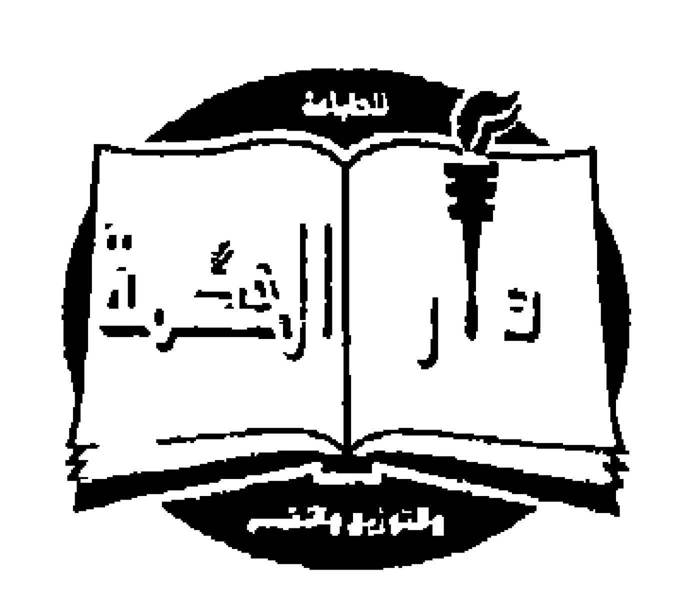


منشورات دار الحكمة
طرابلس - ليبيا

مقدمة إلى البرمجة
بلغة سي

بشير علي القائد
أستاذ مشارك – قسم الحاسب الآلي
كلية العلوم – جامعة طرابلس

الطبعة الخامسة 2012



الطبعة الأولي 1995
الطبعة الثانية 1998
الطبعة الثالثة 2001
الطبعة الرابعة 2007
الطبعة الخامسة 2012

دار الحكمة للطباعة والنشر والتوزيع
هاتف: 3606610 - 3606571
فاكس: 3606610
E-mail: daralhikma_bookshop@yahoo.com
ص.ب 13282
طرابلس – ليبيا

بسم الله الرحمن الرحيم

﴿ سَنُرِيهِمْ آيَاتِنَا فِي الْآفَاقِ وَفِي أَنفُسِهِمْ حَتَّىٰ يَتَبَيَّنَ لَهُمْ أَنَّهُ الْحَقُّ ﴾

صدق الله العظيم

الآية (52) من سورة فصلت

مقدمة الطبعة الخامسة

بسم الله الرحمن الرحيم

الحمد لله الذي علم بالقلم علم الإنسان ما لم يعلم ، والصلاة والسلام على أشرف المرسلين سيدنا محمد وعلى آله وصحبه المهتدين ، وبعد .

فقد قيل قديماً " لا يزال الرجل عالما ما طلب العلم فإذا ظن انه علم فقد جهل ".

بعد نفاد الطبعات الاربعة من هذا الكتاب وبتوفيق من الله تعالى ، ها نحن نقدم طبعته الخامسة شاكرين الله تعالى على تهيئته السبيل لإنجاح الطبعات السابقة ، فقد لاقى هذا الكتاب إقبالا واضحا ونفدت طبعاته بسرعة غير متوقعة ، وهذا يدل على رغبة الإخوة القراء وتطلعهم للمزيد من معرفة إحدى لغات الاتصال مع أجهزة الحاسب الآلي.

وقد قمنا بمراجعة هذا الكتاب للمرة الخامسة مراجعة دقيقة وتعتبر هذه الطبعة منقحة حيث تحتوى على مزيد من المعلومات المنهجية المتعلقة بهذه اللغة الحاسوبية المهمة وكذلك مزيد من التمرينات التي تمكن الدارس من التمرس بهذه اللغة قبل استخدامه لها في الحياة العملية . إلا أننا نحب أن يضع الدارس نصب عينيه أثناء دراسته لهذه اللغة ضرورة أن يتحلى بروح المثابرة والمبادرة وذلك بمحاولته المستمرة لكتابة برامج من إنشائه وتطبيقها

على الجهاز وذلك في كافة المواضيع التي يتم دراستها حتى يتمكن من تعزيز الدراسة الأكاديمية التي يتلقاها بقدر كاف من التطبيق العملي قبل مواجهته للحياة العملية .

وختاماً فإنني أسأل الله الكريم ان يجعل هذا العمل خالصا نافعا باقيا لي ولوالدي ولأهل بيتي وأن يجازي كل من أسهم في مراجعة هذا الكتاب ونشره وإخراجه إلى الوجود في صورة ناجحة شكلا وموضوعا الجزاء الأوفر.

بشير علي القائد
تاجوراء – ليبيا

المحتويات

Contents

5	Preface	المقدمة
7	Contents	المحتويات

الفصل الاول : اساسيات لغة سي

15	Characters	1.1) الرموز
15	Reserved Words	2.1) الكلمات المحجوزة
15	Identifiers	3.1) المعرفات
16	Comments	4.1) التعليقات
17	Numbers	5.1) الأعداد
20	Strings	6.1) السلاسل
20	Character	7.1) الحرف
21	Variables	8.1) المتغيرات
22	Integer Variables	1) متغيرات صحيحة
24	Float Varibles	2) متغيرات حقيقية
24	Double Variables	3) متغيرات الدقة المضاعفة
25	Character Variables	4) متغيرات من النوع الحرفي
25	String Variables	5) متغيرات من النوع السلسلة الحرفية
26	Exercises	9.1) تمرينات

الفصل الثاني : إدخال وإخراج البيانات

1.2) الشكل العام للبرنامج Program Style 27

2.2) دالة الإخراج printf() 28

3.2) مواصفات التحويل Conversion Specifiers 29

4.2) رموز الهروب Escape Characters 30

5.2) توصيف المخرجات Output Formatted 34

6.2) دالة إدخال البيانات scanf() 37

7.2) توصيف المدخلات Input Formatted 38

8.2) تمرينات Exercises 47

الفصل الثالث: الجمل والمؤثرات

1.3) التعبيرات Expressions 51

1) التعبيرات الحسابية Arithmetic Expressions 51

2) التعبيرات المنطقية Logical Expressions 51

2.3) الثوابت Constants 52

3.3) الجمل Statements 53

1) جملة التخصيص Assignment Statement 53

2) جملة التخصيص الممتدة Extended Assignment Statement 56

4.3) المؤثرات Operators 57

1) المؤثرات الحسابية Arithmetic Operators 57

2) المؤثرات العلائقية Relational Operators 60

3) المؤثرات المنطقية Logical Operators 62

4) مؤثرات الزيادة والنقصان Increment And Decrement Operators 63

5) المؤثرات المركبة .... Compound Operators 66

5.3) اولويات تنفيذ المؤثرات .... Operators Precedence 68

6.3) تحويل النوع .... Type Conversions 69

7.3) تمرينات .... Exercises 72

الفصل الرابع: الاختيارات والتبديل

1.4) جملة إذا .... if Statement 75

2.4) الجملة المركبة .... Compound Statement 77

3.4) جملة إذا ... وإلا .... if-else Statement 81

4.4) جملة إذا المتداخلة .... Nested if Statement 83

5.4) جملة التبديل .... switch Statement 87

6.4) تمرينات .... Exercises 98

الفصل الخامس : جمل التكرار

1.5) التكرار .... Repetition 101

2.5) جملة .... while 101

3.5) جملة while المتداخلة .... Nested while Statement 105

4.5) جملة .... do-while 112

5.5) جملة .... for 115

6.5) جملة for المتداخلة .... Nested for Statement 122

7.5) تمرينات .... Exercises 129

الفصل السادس : الجمل التفريعية

1.6) جملة goto 135
2.6) جملة break 137
3.6) دالة exit() 138
4.6) جملة continue 139
5.6) تمرينات Exercises 145

الفصل السابع : دوال التعامل مع الحرفيات

1.7) المؤشر الحرفي Character Pointer 147
2.7) دوال إدخال الحروف Functions For Input Characters 150
1) دالة getch() 150
2) دالة getche() 152
3) دالة getchar() 153
3.7) دوال إخراج الحروف Functions For Output Characters 155
4.7) دوال إدخال وإخراج الحرفيات Input Output String 156
1) دالة gets() 156
2) الدالة puts 157
5.7) دوال معالجة الحرفيات String Manipulation Functions 158
1) دالة القياس strlen Function 158
2) دالة الوصل strcat Function 159
3) دالة الوصل n حرف strncat Function 160
4) دالة النسخ strcpy Function 161
5) دالة نسخ n حرف strncpy Function 162

6) دالة المقارنة .... strcmp Function 164
7) دالة مقارنة n حرف .... strcncmp Function 166
6.7) دوال تبديل الحرفيات .... String Alteration Functions 167
7.7) دوال معالجة الحرف .... Character Manipulation Functions 170
1) دوال اختبار الحرف .... Character Testing Functions 171
2) دوال تبديل الحرف .... Character Alteration Functions 175
8.7) تمرينات .... Exercises 178

الفصل الثامن: مؤثرات الفاصلة والشرطي البت

1.8) مؤثر الفاصلة .... The Comma Operator 181
2.8) المؤثر الشرطي .... The Conditional Operator 185
3.8) الأنظمة العددية .... Numbers System 188
1) النظام العشري .... Decimal System 188
2) النظام الثنائي .... Binary System 188
3) النظام الثماني .... Octal System 189
4) النظام الستة عشري .... Hexadecimal System 189
4.8) مؤثرات البت .... Bitwise Operators 191
5.8) مؤثرات الإزاحة .... Shift Operators 193
6.8) تمرينات .... Exercises 201

الفصل التاسع : الدوال والمؤشرات

1.9) تعريف الدالة .... Function Definition 203
2.9) ملاحظات عن الدالة .... Remarkes on Function 204

3.9) الدالة الفارغة .... void function 205
4.9) جملة .... return 206
5.9) المتغيرات المحلية .... Local Variables 208
6.9) المتغيرات الخارجية .... External Variables 220
7.9) المتغيرات الساكنة .... Static Variables 223
8.9) المؤشرات .... Pointers 225
9.9) الدوال والمؤشرات .... Functions and Pointers 227
10.9) تمرينات .... Exercises 236

الفصل العاشر: دالة الإعادة والماكرو والدوال الرياضية

1.10) دالة الاعادة الذاتية .... Recursion Function 241
2.10) المعالج الأولي .... Preprocessor 249
3.10) الماكرو .... Macro 250
4.10) الدوال العامة .... General Functions 254
5.10) الدوال الرياضية .... Mathematical Functions 257
6.10) تمرينات .... Exercises 261

الفصل الحادي عشر : المصفوفات

1.11) المصفوفة ذات البعد الواحد .... One-dimensional Arrray 265
1) المصفوفة العددية .... Array Numbers 265
2) المصفوفة الحرفية .... Characters Array 276
2.11) المصفوفات ذات البعدين .... Tow-dimensional Arrays 279
1) المصفوفة العددية .... Array Numbers 279

2) المصفوفة الحرفية .... Characters Array 281

3.11) القيم المبدئية .... Initial Values 290

4.11) تمرينات .... Exercises 294

الفصل الثاني عشر : المصفوفات والدوال والمؤشرات

1.12) المصفوفات والدوال .... Arrays and Functions 297

2.12) مصفوفات ذات البعد الواحد والدوال

One-dimensional Arrays and Functions 297

3.12) مصفوفات ذات البعدين والدوال

Two-dimensional Arrays and Functions 303

4.12) المصفوفة والمؤشرات .... Array and Pointers 307

5.12) المصفوفة ذات البعد الواحد والمؤشرات

One-dimensional Array and Pointers 307

6.12) المصفوفة ذات البعدين والمؤشرات

Two-dimensional Array and Pointers 312

7.12) تمرينات .... Exercises 322

الفصل الثالث عشر: النوع والاتحاد والتراكيب

1.13) استخدام النوع .... Typedef 327

2.13) الاتحاد .... Umion 331

3.13) التراكيب .... Structures 336

4.13) التراكيب والدوال .... Structures and Functions 340

5.13) التراكيب والمؤشرات .... Structures and Pointers 344

6.13) تمرينات .... Exercises 357

الفصل الرابع عشر : الملفات

1.14) تعريف الملف .... File Definition 361
2.14) انشاء الملف .... Creating a File 361
3.14) الملفات النصية .... Text Files 363
1) الكتابة في الملف .... Writing in a File 363
2) الاضافة إلى الملف .... Appending to a File 365
3) قراءة الملف .... Reading a File 366
4.14) الملفات الثنائية .... Binary Files 373
1) الكتابة في الملف .... Writing in a File 373
2) قراءة الملف .... Reading a File 374
5.14) الوصول إلى البيانات .... Access to Data 378
1) الوصول التتابعي .... Sequential Access 378
2) الوصول المباشر .... Direct Access 378
6.14) تمرينات .... Exercises 393

ملاحـــق

ملحق (1) جدول أولويات تنفيذ العمليات 396
ملحق (2) جدول الألوان 396
ملحق (3) جدول رموز آسكي ascii 397

1

الفصل الأول

أساسيات لغة سي

تستخدم في لغة C مجموعة من العناصر الأساسية كما هي الحال في أي لغة أخرى وهذه العناصر :-

1.1) الرموز Characters

وهي تتألف مما يلي :-

1- الأرقام (Digits) وهي 0 ,1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9

2- الحروف الهجائية (Letters) الكبيرة A ,B ,C, ..., X, Y, Z أو الصغيرة a ,b ,c, ..., x, y, z

3- الرموز الخاصة (Special Characters) ومنها إشارة الجمع (+) وعلامة الاستفهام (?) والشارحة (:) وغيرها .

2.1) الكلمات المحجوزة Reserved Words

وهي تعني أن هناك بعض الكلمات لها معنى قياسي ، ولا يمكن استعمالها كمتغيرات لأنه يحدث بسببها إرباك للمترجم (Compiler) ومنها :-

```
auto      case      break    char     do
double    else      float    for      goto
if        int       long     return   short
sizeof    struct    static   switch   typedef
union     unsigned  void     while
```

3.1) المعرفات Identifiers

المعرف هو ذلك الاسم الذي يخزن به قيم المتغيرات مثل الثابت أو المتغير

أو الدالة ، ومن شروط المعرف :-

1) أن يتكون من حرف أو مجموعة حروف أو حروف وأرقام أو علامة under score (_) بأي ترتيب .
2) ينبغي أن يبدأ المعرف بحرف أو بالعلامة (_) من الجهة اليسرى .
3) يجب أن يكون خاليا من الرموز الخاصة .
4) يسمح باستخدام الحروف الصغيرة والحروف الكبيرة .

ويمكن أن يكون للمعرف الطول المناسب ولكن يجب أن يكون واضحا وذا معني ومدلول .

مثال (1-3-1)

المعرفات التالية هي معرفات مقبولة

condenser   This_is_an_Example_using_while   area5

على حين أن المعرفات الآتية غير مقبولة ، للأسباب المبينة قرين كل منها:

تبدأ بالرمز الخاص & .... &address

الخانة الأولى رقم وليست حرفا 9digits

بها رمز الخاص # .... CUSTOMER#

لا يسمح باستخدام الفراغ .... first name

كلمة محجوزة .... long

4.1) التعليقات *Comments*

وهي عبارة عن بعض الأوامر الإيضاحية ولا يكون لها أي تأثير لأنها لا تعتبر جزءا من البرنامج وتستخدم لتسهيل إعادة قراءة البرنامج أو تعديله من طرف المبرمج أو الآخرين، وتستخدم التعليقات أيضاً لشرح وبيان السبب

ورآء أي شيء تفعله بالبرنامج وتوضع في أي مكان من البرنامج ، ومن الممكن أن لا تكون موجودة فيه ، ويبدأ التعليق بالرمزين (*/) وينتهي بالرمزين (/*) .

مثال (1-4-1)

يمكن كتابة التعليق بسطر

```
/* THIS IS THE COMMENT STATEMENT */
```

أو بسطرين

```
/* THIS IS THE COMMENT
   STATEMENT          */
```

5.1) الأعداد *Numbers*

العدد هو ذلك المقدار الثابت الذي لا تتغير قيمته ويتكون من مجموعة من الأرقام (Digits) ذات حد أدني وحد أقصى ، تعتمد على نوع المعالج المستعمل ، وهناك نوعان من الأعداد :-

1) الأعداد الصحيحة I*nteger* N*umbers*

هي الأعداد الصحيحة أيا كانت ، موجبة أو سالبة أو حتي صفرا بشرط:-

I) ألا يحتوي على نقطة عشرية أو أسية.

II) العدد السالب يجب أن يسبق بإشارة (-) لبيان أنه سالب.

III) لا يكون فيه أي رمز خاص أو أي حرف هجائي.

مثال (1-5-1)

الأعداد التالية أعداد صحيحة :-

0 -1111 8888 5006

على حين أن الأمثلة التالية غير مقبولة للأسباب المذكورة أمام كل منها

لوجود رمز خاص (,)	71,000
لوجود نقطة عشرية (.)	1.69

وعموماً تنقسم الأعداد الصحيحة حسب السعة المخصصة لكل نوع ومنها:-

I) العدد الصحيح int حيث يخصص له 16 بت (Bit) أو 32 بت يعتمد على نوع المترجم .

II) العدد الصحيح القصير short تخصص له 16 بت .

III) العدد الصحيح الطويل long وتخصص له 32 بت أو 64 بت يعتمد على المترجم .

فمثلا عندما يكون سعة الذاكرة قليلة نستعمل short فيحجز المترجم (Compiler) مساحة أقصر لـ short من int ، على حين نستعمل long عندما نريد حجز مساحة أكبر من int لتخزين عدد صحيح .

هناك أعداد صحيحة بدون إشارة (Unsigned Numbers) وهي الأعداد الصحيحة الموجبة من صفر إلى 65535 التي لا تحتوي على النقطة العشرية أو الأسية، وتستهلك نفس عدد الخانات في الذاكرة كالأعداد الصحيحة (Integer Numbers) ولكن تختلف عنها في المدى ، حيث يتراوح مداها ما بين (32768-) في الحد الأقصي للأعداد السالبة و (32767) في الحد الأقصى للأعداد الموجبة .

وفيما يلي أنواع البيانات التي تستخدم مع الأعداد الصحيحة :-

int short long unsigned
unsigned short unsigned long

2) **الأعداد الحقيقية** **F*loat Numbers***

وهي تلك الأعداد التي بها كسور عشرية ، أي تحتوي على نقطة عشرية، ولا يكون فيها أي رمز أو حرف ، وقد يكون العدد موجبا أو سالبا بوضع إشارة (-) قبل العدد ، والأعداد الآتية هي أعداد حقيقية :

4.45 -11.005 63. 0.009

على حين أن الأمثلة التالية غير صحيحة للأسباب المبينة أمام كل منها :-

لوجود علامة الدولار ....100.50$

لوجود أكثر من نقطة عشرية.............................998.77.4

لعدم وجود نقطة عشرية...555

من الناحية الأخرى يمكن تمثيل العدد الحقيقي بشكل قوة أسية باستخدام حرف (e) أو حرف (E) حيث يدل هذا الحرف على القوة .

وعلى ذلك يمكن تقديم العدد 12.3 على النحو التالي :-

1230.E-2 أو .00123E4 أو 12.3E0 أو 1.23E1

والعدد الحقيقي يأخذ 32 بت ، ويتراوح المدى من 3.4E-38 في الحد الأدنى و 3.4E+38 في الحد الأقصى للأعداد الموجبة و 3.4E-38- في الحد الأدنى و 3.4E+38 في الحد الأقصى للأعداد السالبة .

هناك أيضاً الأعداد ذات الدقة المضاعفة (Double) ، وهي ناحية أخرى لتقديم العدد من حيث عدد الخانات ، وهي 64 بت ويتراوح المدى من

1.7E-308 في الحد الأدنى و308 +1.7E في الحد الأقصى للأعداد الموجبة و1.7E-308- في الحد الأدنى و 1.7E+308 للأعداد السالبة .

6.1) السلاسل *Strings*

من خصائص السلسلة أنها مجموعة من الرموز سواء كانت حروفا أو أرقاما أو رموزا خاصة أو خليطا منها بشرط أن تكون موضوعة بين علامتي التنصيص المزدوجة (") ومن الممكن أن تكون حروفا صغيرة أو كبيرة على حدٍ سواء .

مثال (1-6-1)

فيما يلي بعض الامثلة على السلاسل

" go to ROOM 45 "   " EMPLOYEE NAME "

" 5*2/x+3 "   "mouse"

"ما اسمك ؟ "   " أدخل قيمة الضريبة "

7.1) الحرف *Character*

هو حرف أو رقم أو رمز موضوع بين علامتي التنصيص المفردة (') .

مثال (1-7-1)

الآتي هي بيانات من نوع الحرف :-

'a'   'A'   '7'   '&'

ويجوز أن يكون الحرف بلا إشارة (Unsigned)، ففي هذه الحالة يكون مداه من الصفر إلى 255 ، أما إذا كان بإشارة (Signed) فيكون مداه من) - 128(إلى (127) .

وهذا يعطي لغة C ميزة خاصة حيث يمكن تخصيص عدد لمتغير من نوع الحرف .

وفيما يلي جدول يبين بعض أنواع البيانات ومدى الأعداد التي يمكن تخزينها لكل نوع .

النوع	سعة التخزين بالبايت	المدى
int	16 or 32	32767 to -32768
short	16	32767 to -32768
long int	32 or 64	2147483648 to 2147483647
unsigned long int	32	-21474836748 to 147483647
unsigned short int	16	0 to 65535
float	32	3.4E-38 to 3.4E+38
double	64	1.7E-308 to 1.7E+308
char	8	-128 to 127
unsigned char	8	0 to 255

8.1) المتغيرات *Variables*

هي أسماء رمزية يخصص لها أماكن تخزين في ذاكرة الحاسب حيث يمكن تحويل وتغيير قيمتها من حين إلى آخر وبالتالي إمكانية الرجوع إليها عن طريق هذه الأسماء وذلك أثناء تنفيذ البرنامج .

في لغة C يجب أن يعلن عن المتغيرات صراحة مسبقا ، وإلا فلن يعترف بها المترجم (Compiler) .

الشكل العام لتعريف المتغير هو :-

```
type variable_list ;
```

حيث

variable_list يمثل قائمة بأسماء المتغيرات .

type تعني نوع المتغير الذي يجب أن يكتب بالحروف الصغيرة ويمكن أن تكون من الأنواع الأتية:-

```
int or long or unsigned char or
char or short or unsigned short or
unsigned or unsigned long or
float or double
```

وفيما يلي بعض الأمثلة توضح كيفية الإعلان والتعامل مع المتغيرات :-

1) متغيرات صحيحة *Integer Variables*

وهي التي يخزن فيها العدد الصحيح القصير (short) سواء كان العدد صحيحا سالبا أو موجبا ، ومدى هذا النوع من 32768- إلى 32767 ويجب أن يعلن عن نوع المتغير قبل استعماله بالحروف (int) .

مثال (1-8-1)

الإعلان الآتي يوضح أن كلا من المعرفات y, x, b, a هي متغيرات صحيحة.

```
int a,b,x,y;
```

ويمكن إشهار المتغيرات الصحيحة ، التي يخزن فيها العدد الصحيح الطويل (long) حيث مداه أكبر من مدي العدد الصحيح القصير وذلك بإشهاره بالعبارة long أو long int .

مثال (1-8-2)

الإشهار

```
long a,b,c;
```

يعني أن المتغيرات c, b, a لها سعة كبيرة حيث مدي كل منها يتـراوح مـن -2147483648 إلـى 2147483647 ونلاحظ هنـا الفرق بين مـدى الأعـداد القصيرة والأعداد الطويلة .

مثال (1-8-3)

خذ مثلا الإعلان الآتي :-

```
int i, j, k;
i = 200000000;
j = 500000000;
k = i+j;
```

فيه تم تخصيـص قيم عدديـة صحيحـة طويلـه للمتغيرين j, i وتخصيـص مجموعهما للمتغير k وبالتالي إذا ما تمت طباعة قيم هذه المتغيرات بالتشـكيل (%d) سنحصل على الآتي :-

I=-15872 J=25856 K=9984

نلاحظ أن الناتج غير صحيح ، وذلك لأن كلا من المتغيرات k, j, i تـم إشـهارها على أنها متغيرات صحيحة من النوع int وعليه لايمكن تحميل قيمـة كبيرة فيها ، وهذا ينتج عنه عدد فـائض (integer overflow) ولتصحيـح ذلـك يجب الإعلان عن المتغيرات من النوع الطويل (long) كما يلي :-

```
long i, j, k;
i = 200000000;
j = 500000000;
k = i+j;
```

في هذه الحالة يكون ناتج المتغيرات السابقة بإستخدام التشكيل (%ld) كمـا يلي :-

I=2000000 J=500000000 K=700000000

2) متغيرات حقيقية *Float Variables*

هي التي تحتوي على نقطة عشرية ، أي العدد الذي به قيمة كسرية وينبغي الإعلان عن هذا النوع من المتغيرات بالكلمة (float) .

مثال (4-8-1)

المعرفات c, b, a هي متغيرات حقيقية، خصص لها بعض القيم من نفس النوع .

```
float a, b, c;
a = 5.5555555555;
b = 3.1111111111;
c = a+b;
```

فإذا ما تمت طباعة المتغيرات السابقة بالتشكيل (%.10f) سيكون الناتج هو كالآتي :-

```
A = 5.5555553436
B = 3.1111111641
C = 8.6666660309
```

نلاحظ أن النتيجة لجميع المتغيرات غير التي خصصت والسبب أن جهاز PC، يتحمل عددا من نوع float مقداره ستة أرقام معنوية تقريبا (Six Significant Digits) .

وللحصول على النتيجة السليمة والصحيحة يجب الإعلان عن المتغيرات من النوع الحقيقي الطويل .

```
long float a, b, c;
```

3) متغيرات الدقة المضاعفة *Double Variables*

هي المتغيرات التي تحوي أعدادا صحيحة أو حقيقية ولكنها مضاعفة ، ويعلن عن نوع هذه المتغيرات بكلمة (double) .

مثال (5-8-1)

المعرفات الأتية هي متغيرات مضاعفة

```
double f, m, z;
```

يمكن الحصول على نفس النتيجة في المثال (4-8-1) بإشهار تلك المتغيرات لتصبح مضاعفة الدقة كالآتي :-

```
double a, b, c;
```

4) متغيرات من النوع الحرفي *Character Variables*

هي نوع آخر من المتغيرات التي يمكن الإعلان عنها بالبرنامج بحيث تحمل خانه واحدة فقط .

مثال(6-8-1)

الإعلان الآتي

```
char a, b;
a = '?';
b = '&';
```

تم فيه استخدام الكلمة char لتدل على أن المتغيرين b, a هما من النوع الحرفي مع إسناد الرمزين ? ,& لكل منهما على التوالي .

5) متغيرات من النوع السلسلة الحرفية *String Variables*

مثال (7-8-1)

الإعلان

```
char str[20] = "Welcome my friend";
```

هو إشهار المتغير str من نوع السلسلة الحرفية (String) طولها 20 حرفا مع تخزين السلسلة Welcome my friend بهذا المتغير .

9.1) تمرينات Exercises

1) اشرح ما هو المقصود بالمعرفات مع ذكر بعض منها .

2) حدد صلاحية أو عدم صلاحية كل من الثوابت الآتية مع ذكر نوعها:-

oxdk '//3' 0.126e3 -456 OxAF 56L 012

3) ما هي المعرفات غير المقبولة مع ذكر السبب للآتي :-

Double	Whynot?	-tax-rate	K : 66
Student	name	nice-to-meet-you	5tests
"Quctation"	totalsum	first name	

4) عرف الكلمة المحجوزة مع ذكر ثلاث منها .

5) ما هو المقصود بكلمة comment ؟ ومتي يتم استخدامها مع إعطاء بعض الأمثلة على ذلك .

6) وضح الفرق بين كل من :

a) long and short
b) identifiers and variables
c) double and float
d) string and character

7) أعط بعض الأمثلة عن كل فقرة من فقرات التمرين (6) .

8) حدد أي التعريفات الآتية غير صحيحة مع ذكر السبب.

a) int FLOAT;
b) LONG integer;
c) unsigned float x,y,next;
d) char char1;char2;
e) double density;
f) short way,to_go;
g) long float x and y
h) char string(50);

9) اذكر ما هو الخطأ إذا كان هناك خطأ مع ذكر السبب لما يلي :-

a) /* This is an example /* using */ comment statement */
b) /* This is an example /* using comment statement */
c) //* This is an example using comment ststement *//

2

الفصل الثاني
إدخال وإخراج البيانات

1.2) الشكل العام للبرنامج *Program Style*

يكون الشكل العام للبرنامج في لغة C كالتالي :-

<header files>	ملفات العناوين
<preprocessors>	توضيحات خارجية
<declarations>	إشهارات خارجية
main( )	
{	
<declarations>	إشهارات داخلية
statement(s) ;	جملة أو جمل البرنامج
}	

مثال (2-1-1)

حتى نكون على بينة من تركيب البرنامج بلغة C وجب إعطاء مثال أولي يبين هذا التركيب وهو كالموضح فيما بعد.

```
                    /* THIS IS PROGRAM #1 */
#include <stdio.h>
  #include <conio.h>
main( )
 {
        clrscr();
     printf(" Hi there how are you today? ") ;
 }
```

شرح البرنامج: السطر الأول توجد به جملة

THIS IS PROGRAM #1

محاطة بين (*/) من اليسار و (/*) من اليمين وهذا يعني أن الجملة هي جملة تعليق على البرنامج حيث يتجاهلها المترجم وقت التنفيذ .

يلي ذلك الأمر

```
#include <stdio.h>
```

وهو اختصار للعبارة standard input-output header الذي يسمح للبرنامج باستعمال دالة الإدخال والإخراج الموجودة في الملف <stdio.h> ويسمي ملف عنوان header file ، اما الامر <conio.h> فهو يسمح للامر clrscr() بمسح كل المحتويات على شاشة العرض وقت تنفيذها .

أما الدالة main( ) فهي الدالة الرئيسية التي يبدأ بها أي برنامج يلي ذلك القوس المفتوح } الذي يدل على بداية جسم البرنامج (Program body) .

أما الدالة printf فتقوم بطباعة الرسالة الموجودة بين علامتي التنصيص (") على شاشة العرض وهي :

```
Hi there how are you today?
```

أخيراً من الضروري أن ينتهي البرنامج بالقوس المغلق { للدلالة على نهاية الدالة main() وبالتالي نهاية البرنامج .

2.2) دالة الإخراج *printf*( )

تستخدم هذه الدالة لإخراج البيانات والمعلومات من داخل ذاكرة الحاسب وعرضها على وحدة الإخراج القياسية (Standard Output Unit) والمسماة الشاشة (Screen) وهي تنتمي للأمر

```
# include <stdio.h>
```

الصيغة العامة لهذه الدالة هي

```
printf ("format", arg1, arg2, ...);
```

حيث :

الحرف f المنتهية به الدالة يدل على أنه ذات مواصفات (Formatted) .

و format تعني التوصيف أو التشكيل اللازم للطباعة على أن يوضع بين علامتي التنصيص المزدوجة (") ولتحويل كل الصفات يجب البدء بالرمز (%) والقيمة يحددها الحرف الذي يلي هذا الرمز .

arg1, arg2 هي عناصر البيانات ويمكن ان تكون قيم لمتغيرات أو تعبيرات أو سلاسل أو ثوابت عددية ، المطلوب طباعتها على الشاشة حسب الوصف أو التشكيل (format) على أن تُفْصَلَ هذه العناصر عن بعضها بواسطة فواصل.

3.2) مواصفات التحويل *Conversion Specifiers*

ذكرنا فيما سبق أن الدالة printf() تحتوي على جزء التوصيفات اللازمة لطباعة البيانات ، وعليه نقدم فيما يلي بعض رموز التشكيل والتحكم حيث يتم عرض المتغيرات تبعا للرموز التالية :-

الرمز	المعنى
%c	للحرف (Character)
%s	للسلسلة الحرفية (String)
%d or %i	للعدد الصحيح (Integer)
%e or %E	للعدد الحقيقي بالصورة الأسية (double)
%f	للعدد الحقيقي (float)

الرمز	المعنى
%g or %G	لطباعة العدد بالتوصيف %f أو %e أيهما اقصر
%u	للعدد الصحيح بدون إشارة (Unsigned)
%o	للعدد الصحيح بالنظام الثماني (Octal)
%x or %X	للعدد الصحيح بالنظام الستة عشرى(Hexadecimal)
%p	لقيمة المؤشر (Pointer)
%%	لطباعة العلامة (%)

4.2) رموز الهروب *Escape Characters*

هناك بعض رموز أو حروف الهروب التي تستخدم لأغراض خاصة مع دالة الطباعة printf( ) للتحكم في طباعة المخرجات على شاشة العرض التي تبدأ بالخط المائل (\ ) وتكون ضمن علامة التنصيص المزدوجة (") ، والجدول الآتي يتضمن بعض هذه الرموز .

اسم الرمز	كيفية كتابته
القفز إلى سطر جديد	\n
البدء من أول السطر	\r
التقدم 7 مسافات عمودية قبل الطباعة	\t
الإنتقال إلى صفحة جديدة	\f
استخدام الجرس	\a
القفز مسافة إلى الخلف	\b
لطباعة الحرف (\)	\\
لطباعة علامة التنصيص المزدوجة (")	\"
لطباعة علامة التنصيص الفردية (')	\'

مثال (2-4-1)

الأمر

```
printf("10 IS %d , WHILE 10+15 IS %d" , 10 , 10+15);
```

سيطبع الآتي :-

```
10 IS 10 , WHILE 10+15 IS 25
```

مثال (2-4-2)

خذ مثلا البرنامج الآتي :-

```
#include <stdio.h>
main( )
{
     printf("My name is");
     printf("RAEF BASHIR");
     printf("What is yours ?");
 }
```

عند تنفيذه سيظهر على الشاشة السطر الآتي :-

```
My name isRAEF BASHIRWhat is yours ?
```

الذي يلاحظ أنه بالرغم من وجود ثلاث أوامر لدالة الطباعة ، فالناتج قد ظهر في سطر واحد فقط ، مع عدم وجود فراغ بين هذه الجمل بالأوامر الثلاثة .

مثال (2-4-3)

ولكن ماذا نفعل إذا أردنا كتابة ناتج تنفيذ الجمل الثلاثة السابقة كل واحدة في سطر منفصل ؟ هنا يجب استخدام الرمز الخاص بالانتقال إلى سطر جديد وهو الرمز (\n) داخل علامتي التنصيص .

```
#include <stdio.h>
main()
{
      printf("My name is \n");
      printf("RAEF BASHIR \n");
      printf("What is yours ?");
 }
```

هنا سَتُطْبَعُ العبارة My name is في بداية السطر الأول ، وحيث إن هذه العبارة تنتهي بالرمز (n\) التي ينتج عنها القفز إلى سطر جديد قبل كتابة الجملة الموالية ومن ثم تطبع العبارة RAEF BASHIR في بداية السطر الثاني وحيث إن هذه العبارة تنتهي أيضاً بالرمز (n\) عليه ستطبع العبارة الموالية ? What is yours في السطر الثالث ، عموما سيكون ناتج البرنامج السابق مشابها للآتي :-

```
My name is
RAEF BASHIR
What is yours ?
```

يمكن أيضاً استخدام الرمز (n\) في بداية علامة التنصيص كما يبينه البرنامج الآتي :-

```
#include <stdio.h>
main()
{
      printf("\n My name is ");
      printf("\n RAEF BASHIR\n");
      printf("What is yours ?");
 }
```

الناتج سيكون مشابها للمثال السابق .

مثال (2-4-4) المطلوب كتابة برنامج لإصدار دعوة باسم MIRATH GAYED لحضور حفل يقام يوم 17/2/2012 بالمركز .



لتصميم البرنامج يجب تحديد الرسائل الموجهة بمواصفات الطبع وعلى هذا يمكن أن يكون البرنامج كما يلي :-

```
#include <stdio.h>
main( )
{
     printf("\n Hi MIRATH GAYED\n YOU ARE INVITED ON ");
     printf("17/2/2012 IN THE CENTER");
}
```

وبالتالي تطبع الدعوة بداية من السطر الأول بالشكل الآتي :-

```
Hi MIRATH GAYED
YOU ARE INVITED ON 17/2/2012 IN THE CENTER
```

مثال (2-4-5)

المثال الآتي يبين استخدام بعض رموز الهروب .

```
#include <stdio.h>
main( )
{
     printf("\n1234567890123456789012345678901234567 89\n");
     printf("\r\" THIS IS AN EXAMPLE \"\n");
     printf("\t\tRED \t GREEN \\ BLUE");
}
```

وقت تنفيذ هذا البرنامج سيعطي النتائج التالية :-

```
123456789012345678901234567890123456789
"bTHISbISbANbEXAMPLEb"
bbbbbbbbbbbbbbbREDbbbbbGREENb\bBLUE
```

حيث يعني الحرف (b) أنه فراغ .

5.2) توصيف المخرجات *Output* **Formatted**

مثال (2-5-1)

على فرض أن لدينا التعريفات وجمل التخصيص للمتغيرات الصحيحة والحرفية التالية :-

```
int a,b;
long x = 987654321;
char d = 'A';
a = 5 ; b = -35 ;
```

فيما يلي جدول يوضح التوصيفات مع هذه المتغيرات .

التوصيف	المتغير	الناتج
%d	a	5
%5d	b	bb-35
%ld	-x	-987654321
%-4d	a	5
%05d	a	00005
%c	d	A
%3c	d	bbA
%-4c	d	A
%d	d	65
%12ld	x	bbb987654321

الشرح :

عند التعامل مع المتغيرات الصحيحة والحرفية فإن التخزين يبدأ من اليمين إلى اليسار من خلال الخانات المخصصة للمتغير مع ترك البقية كفراغات عند الزيادة ، بالجدول السابق استُعمِل التوصيف (%5d) مع المتغير b وهذا يعني تخصيص 5 خانات على شاشة العرض لطباعة الرقم المخزن بالمتغير وهو 35- بداية من اليمين مع ترك فراغين على يسار هذا الرقم ، وبنفس الطريقة طبعت قيمة المتغير الحرفي d بالتوصيف (%3c) .

مثال (2-5-2)

على فرض ان لدينا التعريفات وجمل التخصيص للمتغيرات الحقيقية التالية:-

```
float v = 98.4973 ;
double x = 55.123456789 ;
```

فيما يلي جدول يوضح التوصيفات مع هذه المتغيرات .

الناتج	المتغير	التوصيف
55.123457	x	%f
55.12	x	%.2f
5.512346e+01	x	%e
5.512346E01	x	%E
55.123457	x	%lf
55.123456789	x	%.9f
55.123456789	x	%.9lf
b98.497	v	%7.3f
bbb98.4973	v	%10.4f
98.497299194	v	%.9f

عند استخدام التوصيف (%f) مع المتغير x فهذا يعني طباعة قيمة المتغير مع تخصيص 6 خانات فقط للقيمة الكسرية ، في حين أن التوصيف (%10.4) يعني طباعة قيمة المتغير x في حقل طوله لا يقل عن 10 خانات من ضمنها 4 خانات للقيمة الكسرية .

مثال (3-5-2)

يجب أن تكون التوصيفات مطابقة لنوع المتغيرات وإلا فستكون النتيجة خاطئة بطبيعة الحال ، وإليك الآن البرنامج الآتي مع التنفيذ لتبين هذه الحالة .

```
#include <stdio.h>
main( )
{
   float y;
   int x;
   x = 10;
   y = 3.456;
   printf(" X==>%f\n Y==>%d" , x , y);
}
```

الناتج قد يكون مشابها للآتي :-

```
X==>116.736023
Y==> 0
```

كما نلاحظ هنا أن النتائج كانت خاطئة والسبب هو استخدام التوصيف (%f) مع المتغير الصحيح x والتوصيف (%d) مع المتغير الحقيقي y .

مثال (4-5-2)

البرنامج الآتي يوضح استخدام بعض المتغيرات المختلفة مع توصيفاتها لإصدار بطاقة حضور الاجتماع باسم آية بشير .

```
#include <stdio.h>
main()
{
    int day, year;
    char c1, c2, c3;
    char name[] = "AYAH BASHIR";
    day = 5;
    year = 2001;
    c1 = 'M';
    c2 = 'A';
    c3 = 'Y';
    printf("\n%s\n", name);
    printf("\t YOU ARE INVITED ON THE %dth OF " , day);
    printf("%c%c%c %d TO THE MEETING", c1, c2, c3, year);
}
```

في هذا البرنامج تم استخدام متغيرات من النوع الصحيح والحرف والسلسلة مع التوصيف المقابل لكل متغير ، وعند تنفيذه ستظهر الرسالة بالشكل الآتي :-

AYAH BASHIR
YOU ARE INVITED ON THE 5th OF MAY 2001 TO THE MEETING

6.2) دالة إدخال البيانات *scanf*( )

في البرامج السابقة استخدمنا مؤثر التخصيص (=) لإسناد قيم لمتغيرات وهذا لا يسمح بتغيير تلك القيم إلا بتغير جملة الإسناد بحيث تكون ثابتة اثناء تنفيذ البرنامج .

من هنا وجب تقديم دالة scanf() التي تماثل صيغة الدالة printf() حيث إنها معرّفة في المكتبة <stdio.h> وهي تأخذ البيانات عن طريق لوحة المفاتيح وتخصصها لأسماء متغيرات بحيث يمكن إستخدامها في البرنامج فيما بعد .

الشكل العام لهذه الدالة هو

```
scanf("format" , &variable);
```

حيث :

الحرف f المنتهية به دالة الإدخال scanf() يدل على أنه ذو صيغة أو مواصفات معينة (Formatted) .

format يعني التوصيفات المختلفة التى تُحتويها الدالة scanf() وهذا الجزء يجب أن يوضع بين علامتي التنصيص المزدوجتين (") وذلك لتشكيل المتغيرات الداخلة .

variable يعني المتغير المطلوب تخزين قيمة فيه بشرط أن يسبق هذا المتغير الرمز(&) الذي يشير إلى عنوان ذلك المتغير في ذاكرة الحاسب .

خذ مثلا الجملة

```
scanf("%f" , &y);
```

عندما يصل البرنامج إلى هذه الجملة يتوقف منتظرا إدخال عدد حقيقى للمتغير y عن طريق وسيلة الإدخال لوحة المفاتيح ، ثم تخزن تلك القيمة في عنوان هذا المتغير المخصص له في الذاكرة .

وفيما يلي رموز التوصيف للدالة scanf() التي يجب أن تبدأ بالرمز(%) وهي شبيهة برموز التوصيف لدالة الإخراج printf() .

7.2) توصيف المدخلات Formatted Input

فيما يلي التوصيفات الخاصة بادخال البيانات المختلفة :-

الرمز	المعنى
%c	للحرف (character)
%s	للسلسلة الحرفية (String)
%d	للعدد الصحيح (Integer)
%e or %E	للعدد الحقيقي بالصورة الاسية (double)
%f	للعدد الحقيقي (float)
%0	للعدد الصحيح بالنظام الثماني (Octal)
%x	للعدد الصحيح بالنظام الستة عشرى (Hexadecimal)

ملاحظة:

يجب أن يؤخذ في الأعتبار أن الأحرف x,o,e,f,d يمكن :

1) أن يسبقها الحرف h للتحويل إلى فئة short .

2) أن يسبقها الحرف l للتحويل إلى فئة long int أو long float .

مثال (2-7-1)

لنفرض أن لدينا الجزء الآتي من برنامج معين

```
char a;
int m;
float x;
long float z;
scanf("%c  %d  %f  %lf " , &a, &m, &x, &z);
```

في دالة الإدخال scanf() يمكننا قراءة أربعة قيم يتم إدخالها عن طريق لوحة المفاتيح إما في سطر واحد أو اكثر من سطر على أن يفصل بينهم فراغ أو أكثر بحيث تكون :

القيمة الأولى من نوع الحرف للمتغير الحرفي a .

القيمة الثانية من النوع الصحيح للمتغير الصحيح m .

القيمة الثالثة من النوع الحقيقي للمتغير الحقيقي x .

القيمة الرابعة من النوع الحقيقي الطويل للمتغير الحقيقي الطويل z .

مثال (2-7-2)

المطلوب كتابة برنامج مهمته استقبال قيمتين مع إيجاد مجموعيهما .

```
#include <stdio.h>
main( )
{
     int x, y, sum;
     scanf("%d  %d", &x, &y);     /* inputs are separated by a comma  */
     sum = x+y;
     printf("The sum of %d  and %d  is %d", x, y, sum);
}
```

هنا تم الإعلان عن المتغيرات sum, y, x من النوع الصحيح تلا ذلك الدالة scanf() لاستقبال قيمتين من النوع الصحيح ، وعند تنفيذ هذا البرنامج يتم إدخال القيمتين 5, 9 إما بسطر واحد يفصل بينهما فراغ أو أكثر أو بسطرين وبالتالي تخزينهما بالمتغيرين y, x على التوالي ، أخيراً يتم جمع القيمتين

وتخزينهما بالمتغير sum مع تنفيذ دالة الإخراج printf() التي ينتج عنها السطر الآتي :-

The sum of 5 and 9 is 14

مثال (3-7-2)

البرنامج الآتي مهمته استقبال عدد من المتغيرات المختلفة مع إخراجها .

```
#include <stdio.h>
main( )
{
      int  i, j;
      float a, b;
      char x, y;
      scanf("%d %d %f %f %c %c", &i, &j, &a, &b, &x, &y);
      printf("I=%d  J=%d  A=%.2f ", i, j, a);
      printf("\n B=%.2f  X=%c  Y=%c", b, x, y);

  }
```

عند تنفيذه يجب فصل عناصر البيانات العددية عن بعضها بفراغ أو اكثر من فراغ في حالة كون البيانات المدخلة عددية ، أما البيانات من النوع الحرفي فلا يجب ان توضع بين علامتي التنصيص المفردة (') ولا يمكن فصلها عن بعضها بفراغ .

وعلى فرض أن المطلوب إدخال القيم الآتية وتخصيصها للمتغيرات قرين كل منها :-

المتغير	القيمة
i	-999
j	5432
a	567.0
b	-4321.67
x	W
y	Z

عليه يمكن إدخال هذه القيم على النحو الآتي :-

```
-999  5432  567.0  -4321.67WZ
```

حيث فُصِلَت القيم العددية بفراغ اما الحرفية فلم تفصل .
أو إدخالها بالشكل الآتي :-

```
-999  5432  567  -4321.67WZ
```

أي كتابة الرقم 567 بدون الفاصلة العشرية .

وسيكون ناتج تنفيذ البرنامج في كلا الحالتين مشابها للآتي :-

```
I =-999  J=5432  A=567.00
B=-4321.67  X=W  Y=Z
```

أما إذا أُدْخِلَت البيانات على النحو الآتي :-

```
999   5432   567 -4321.67 WZ
```

فالناتج سيكون الآتي :-

```
I =-999  J=5432  A=567.00
B=-4321.67  X=   Y=W
```

وهي نتيجة خاطئة بالنسبة للمتغيرين y, x والسبب وجود فراغ بين القيمة العددية 4321.67- والحرف W ولهذا خُصِّصَ الفراغ الموجود بينهما للمتغير X وخُصِّصَ الحرف W للمتغير Y .

مثال (2-7-4)

انظر إلى البرنامج الآتي:

```
#include <stdio.h>
main( )
{
    int i;
    char a, b;
    scanf("%c %c %d", &a, &b, &i);
    printf("A=%c  B=%c  I=%d", a, b, i);
}
```

وفيه يمكن إدخال القيم الثلاث في إحدى الحالتين :-

1) XY10
2) XY 10

أي عدم استخدام الفراغ بين البيانات من نوع الحرف وآخر قيمة عددية صحيحة في الحالة (1) أو استخدام الفراغ بين القيمة الحرفية وآخر قيمة عددية في الحالة (2) وذلك حين يكون وجود المتغيرات الحرفية قبل المتغيرات العددية

عموماً يكون ناتج هذا البرنامج في الحالتين هو الآتي :-

```
A=X  B=Y  I=10
```

مثال (2-7-5)

ماذا يحدث إذا نسيت أن تضع الرمز (&) قبل المتغير مع الدالة scanf() ؟

الحل : البرنامج الآتي يوضح الجواب .

```
#include <stdio.h>
main()
{
    int x, y;
    scanf("%d  %d", x, &y);
    printf("X=%d  Y=%d", x, y);
}
```

وبإدخال القيمتين 33 ,44 لكل من المتغيرين y, x على التوالي وحيث إن المتغير x لم يسبقه الرمز (&) فعليه لا يمكن الإشارة إلى عنوان ذلك المتغير في الذاكرة وبالتالى ستكون النتيجة غير متوقعة للمتغير x مع رسالة تبين هذا الخطأ كالآتي :-

X=0 Y=44Null pointer assignment

كما عرفنا سابقاً فإنه من الضروري عند استخدام الدالة scanf() أن تكون التوصيفات متطابقة من حيث النوع والعدد .

مثال(2-7-6)

خذ مثلا البرنامج الآتي :-

```
#include <stdio.h>
main( )
{
    int n;
    scanf("%f" , &n);
    printf("N=%d" , n);
}
```

عند تنفيذه ينتج عنه رسالة الخطأ الآتية:-

scanf : floating point formats not linked
Abnormal program termination

والسبب هو عدم توافق نوع المتغير الصحيح n مع التوصيف (%f) المستخدم مع دالة الإدخال scanf() .

في كل البرامج التي كُتِبت في السابق لا توجد إشارة تبين عدد القيم المدخلة أو نوعها من حيث إنها حرف أو سلسلة أو أرقام ، وعلى هذا يجب أن يكون المبرمج على علم بهذه المتغيرات حتى يتم إعطاء البيانات المناسبة

لها عن طريق لوحة المفاتيح وذلك بطباعة الرسائل اللازمة قبل الوصول إلى إدخال البيانات عن طريق الدالة scanf() .

مثال (2-7-7)

البرنامج الآتي هو إعادة للبرنامج الوارد بالمثال (2-7-2) وفيه يتم طباعة الرسالة المناسبة قبل الوصول إلى الدالة scanf() لتبين نوع البيانات المطلوب إدخالها .

```
#include <stdio.h>
main( )
{
      int x, y, sum;
      printf("Enter the first number ==> ?");
      scanf("%d ", &x);
      printf("Enter the second number ==> ?");
      scanf("%d ", &y);
      sum = x+y;
      printf("The sum of %d  and %d  is %d", x, y, sum);
}
```

هنا ستظهر الرسالة الأولى :

Enter the first number ==> ?

على شاشة العرض والتي تسأل عن إدخال الرقم الأول مع إيقاف مؤشر الشاشة بعد علامة الاستفهام (?) مباشرة ، وللإجابة عن هذا السؤال يُطْبَعُ الرقم وليكن 5 ثم يُضْغَطُ على مفتاح الإدخال Enter عندها تظهر الرسالة الثانية :

Enter the second number ==> ?

وبنفس الطريق يتم الرد عليها بإدخال الرقم 9 وبالتالي يرد البرنامج بالجواب الآتي :-

The sum of 5 and 9 is 14

مثال (2-7-8)

باستخدام الدالة scanf() والدالة printf() المطلوب كتابة برنامج مهمته قراءة البيانات الاسم واللقب والعمر.

```
#include <stdio.h>
#include <conio.h>
main()
{
    char first[15], last[15];  float age;
    clrscr();
    printf("\nWhat is your first name ? ");  scanf("%s", first);
    printf("What is your last name ? ");     scanf("%s", last);
    printf("How old are you ? ");     scanf("%f", &age);
    printf("My name is %s %s ", first, last);
    printf("and I am %.1f years old.", age);
}
```

بعد الإعلان عن المتغيرين last, first من نوع السلسلة الحرفية حجم كل واحدة منها 15 حرفا ، ثم إدخال الاسم واللقب باستخدام التوصيف (%s) والعمر بالتوصيف (%f)،عليه عند تنفيذ هذا البرنامج سينتج عنه مسح شاشة العرض ثم نوع من التحاور بين البرنامج ومستخدمه تتمثل في الآتي :-

```
What is your first name ? ANAS
What is your last name ? GAYED
How old are you ? 13.5
```

وأخيراً يكون الرد هو :

```
My name is ANAS GAYED and I am 13.5 years old.
```

مثال(2-7-9) المطلوب كتابة برنامج لقراءة أربعة قيم حقيقية مع إيجاد متوسطهم .

```
#include <stdio.h>
main( )
{
        /* Write a program to input four real
                numbers and output their average  */
    float n1, n2, n3, n4, sum, avg;
    printf("Enter 4 Real Numbers Please ==>  ");
    scanf("%f  %f  %f  %f", &n1, &n2, &n3, &n4);
    sum = n1+n2+n3+n4;
    avg = sum/4;
    printf("The Average of %.2f  , %.2f ", n1, n2);
    printf(" , %.2f  , %.2f ==> %.3f ", n3, n4, avg);
}
```

التنفيذ يعطي النتائج المشابهة للآتي :-

```
Enter 4 real Numbers Please ==> 6.80  1.34   0.05  45.2
The Average of  6.80 , 1.34 , 0.05 , 45.20 ==> 13.347
```



8.2) تمرينات Exercises

1) اجعل الحاسب يطبع اسمك كاملا بداية من العمود التاسع والسطر السادس
2) المطلوب كتابة برنامج مهمته طباعة اسمك كاملا في السطر الثالث، وعنوانك في السطر السادس ، وعمرك في السطر الثامن باستخدام دالة الطباعة ()printf واحدة فقط .
3) اكتب جمل الإشهار المناسبة لرقم الدواء وسعره وتاريخ صلاحيته .
4) باستخدام الاشهار بالتمرين (3) ، المطلوب قراءة هذه البيانات مع طباعتها بالشكل المناسب .
5) اكتب برنامجا لقراءة أطوال أضلاع مستطيل مع حساب :
   محيطه = مجموع أضلاعه الأربعة .
   مساحته = الطول X العرض .
6) أوصف ناتج الآتي :-

```
a) printf("a=%05d ", 555);
b) printf("b=%5s", "abc");
c) printf("c=%E", 23.68956);
d) printf("d=%.2f", 23.63936);
e) printf("e=%.0f", 23.61234);
f) printf("f=%ld", 2368956);
g) printf("g=%d", 0<= ! (9));
h) printf("h=%11.2f", 94.5678);
i) printf("i=%=7.4f", 94.5377);
j) printf("j=%2c", '&');
k) printf("k=%8d", -4321);
l) printf("l=%d", 'A'>'a');
m) printf("m=%2c", "&");
n) printf("n=%f", 3456.4321);
```

7) افحص الأخطاء مع التصحيح إن وُجِدت في البرامج الآتية :-

a)

```
main ()
 {
        int a,y, float b,
        scanf("%D" ; A);
        printf("%f" , b);
        printf("/n/n")  y=A+b;
        printf("A= %d  ,  B=  %d, Y=%d,a,b,y");
 }
```

b)

```
main()
{
        int  x , float y ,x:;     x= 45.0;  X=9876543;
        scanf("A=%f " , x);
        printf("x+y=%d", &x,&y);
}
```

8) استخدم الورقة والقلم أولاً ثم جهاز الحاسب ثانياً لإيجاد الناتج للآتي :-

a) printf("\n%.0f plus %.2f < %.3f ", 25.86 , 16.013 , 50.5662);
b) printf("\nA*B ====>%ld", 2000 * 1000);
c) printf("\nSUB\nTRAC\nTION \n %d " , 35-8);
d) short M= -9; printf("\nM equal to %d ", -M);

9) بعد إدخال القيم 6 ,5 ,66 ,55 عن طريق الدالة scanf() ، اكتب دالة printf() واحدة ينتج عنها إظهار البيانات السابقة بالشكل الآتي :-

THE SUM OF 66 AND 55 = 121
THE PRODUCT OF 6 TIMES 5 IS EQUAL TO 30

10) أكتب برنامجاً يقوم بإدخال التاريخ 17-2-2011 بحيث يتم قراءة اليوم والشهروالسنة بالمتغيرات day,manth,year مع أظهارها بالصورة الآتية

Our date is 17/2/2011

11) قم بكتابة برنامج يقوم باستقبال رقم قيد الطالب ودرجاته في ثلاثة أمتحانات التي قد تكون على النحو التالي:-

Type student ID number please ==>: 17022011

Type first test please ==>: 69.5

Type second test please ==>: 71

Type third test please ==>: 80

ثم طباعة البيانات السابقة مع المتوسط بالشكل الموالي:-

Student ID# is 17022011

Test 1 ==> 69.5

Test 2 ==> 71.0

Test 3 ==> 80.0

Average ==> 73.5

12) اعد كتابة البرنامج بالتمرين السابق على ان تكون النتائج بالشكل التالي :-

Student Report

Id Number	test_1	test_2	test_3	total	Avg
17022011	69.5	71.0	80.0	220.5	73.5

13) على فرض انك أعطيت الأشهار بالشكل التالي :-

```
long a=-77665544;   float b=765.56713;
 int  x=8764;   char c='R';
```

صف ناتج كل فقرة من الفقرات الآتية :-

```
a) printf("\nA1=%9ld  B1=%f", a,b);
c) printf("\nA3=%11ld  B3=%10.3f", a,b);
d) printf("\nA4=%8ld  B4=%.0f", a,b);
e) printf("\nA5=%11ld  B5=%10.3f", a,b);
f) printf("\nB6=%E  B7=%.2f", b,b);
g) printf("\nX1=%07d X2=%2d C=%4c", x,x,c);
```

14) اذا ما اعطيت الاعلانات التالية :-

```
float X;  long Y;  int A; char B,C;
```

والبيانات التي يمثلها السطر التالي :-

123456789 123.4567

أوجد ناتج دالتي الإدخال والإخراج في كل فقرة من الفقرات الآتية :-

```
a)  scanf("%f%d%c",&X,&A,&C);
    printf("\nX=%f  A=%d  C=%c",X,A,C);
b)  scanf("%c%c%f%d",&B,&C,&X,&A);
    printf("\nB=%3cC=%cX=%fA=%d",B,C,X,A);
c)  scanf("%ld%c%f",&Y,&C,&X);
    printf("Y=%ld  X=%.0fC=%c",Y,X,C);
d)  scanf("%ld%f",&Y,&X);
    printf("\nY=%ld  X=%f",Y,X);
```

15) قم بكتابة برنامج مهمته قراءة قيمتين Y,X مع إيجاد حاصل جمعهما وضربهما على ان يكون الناتج بالصورة التالية :-

X	Y	X+Y	X*Y
...	...	...	...

16) اكتب برنامجا يقرأ درجة الحرارة بالفهرنهية ثم يطبعها بالمئوية مستخدماً المعادلة :

$$C = \frac{5}{9} (F - 32)$$

3

الفصل الثالث

الجمـــــل والمـــؤثرات

1.3) التعبيرات *Expressions*

التعبير يمكن أن يكون قيمة ثابتة مثل 66.7 أو متغيراً مثل x أو متغـيرا مـن نوع المصفوفة مثل matr[5] أو مجموعـة مـن القيـم والمتغيرات متصلـة مـع بعضـها بواسطة مؤثر (Opreator) أو أكثر ومنها :-

1) التعبيرات الحسابية *Arithmetic Expressions*

هي مجموعة من المتغيرات أو الثوابت أو الاثنين معا بحيث تكون نتيجتها قيمة حسابية صحيحة أوحقيقية.

خذ مثلا التعبير

((5+a)*b-2.7)+c

فيه c, b, a متغيرات و 5, 2.7 ثوابت على حين - ,* ,+ مؤثرات حسابية .

2) التعبيرات المنطقية *Logical Expressions*

تتكون من متغيرات أو قيم وعلاقة منطقية واحدة أو أكثر ، حيث تعطـي نتيجة إحدى القيمتين المنطقيتين 1 ويعنـي صحيـح (True) أو 0 ويعنـي خطـأ (False) .

مثال لتعبيرين منطقيين

```
Grade >= 50
z != x
```

2.3) الثوابت *Constants*

الثابت يحمل قيما لا يمكن تغييرها في البرنامج و تستخدم الثوابت في لغة C حيث يعلن عنها في بداية البرنامج مرة واحدة وقبل أن تظهر أي جملة من جمل لغة C ويدل الثابت على نوع المتغير الذي به القيمة.

والصيغة العامة للثابت هي:

```
const  type  name = value;
```

حيث name هو معرف يمثل اسم الثابت و type نوعه في حين value هو عنصر البيانات الفعلي الذي يحدد له المعرف name .

مثال (3-2-1)

فيما يلي بعض الأمثلة المختلفة على الثابت :

```
const  char  *department = " Computer Sceince";
const  char ch = '?';
const  int DaysPerWeek = 7;
```

مثال (3-2-2)

البرنامج الآتي يبين كيفية استعمال مؤثر const مع بعض المعرفات المختلفة.

```
#include <stdio.h>
main( )
{
    const  a = 0xB;
    const char *string  = " This an example to use a constant ";
    const char *line = " ------------------------------------------------ ";
    printf("\n %s \n %s", string, line);
    printf("\n a in hexadecimal = %d in decimal ", a);
    printf("\n\t\t  = %o in octal ", a, a);
    printf("\n\t\t  = %x  in hexadecimal ", a);
}
```

في البرنامج وعن طريق الثابت .const تم إسناد الرقم 0xB بالنظام الستة عشرى للمعرف a الذي يكافئ القيمة 11 بالنظام العشري الذي سيأتي شرحه فيما بعد إن شاء الله وكذلك تخصيص سلسلتين حرفيتين إلى معرفات ,line string وعند تنفيذ هذا البرنامج سنحصل على النتائج المشابهة للآتي :-

```
This an example to use a constant
------------------------------------------
a in hexadecimal = 11 in decimal
                 = 13 in octal
                 =  b in hexadecimal
```

3.3) الجمل *Statements*

الجملة هي الإشارة أو الأمر الموجه إلى جهاز الحاسب الآلي لعمل الشيء المطلوب القيام به ويكتب الأمر إما في سطر واحد أو في اكثر من سطر.

وتكون الجملة إما مفردة وهي عبارة عن أمر أو تعليمة واحدة في حين الجملة المركبة هي التي تتكون من أكثر من جملة مفردة على أن تكون محصورة بين القوسين { } وفي كلا الحالتين يجب أن تنتهي كل جملة بالفاصلة المنقوطة .

1) جملة التخصيص *Assignment Statement*

جملة التخصيص تعني تخصيص أو إسناد قيمة إلى متغير وهي تأخذ الشكل العام الآتي :-

```
Variable = Expression ;
```

حيث الطرف الأيسر من رمز التخصيص (=) دائماً يكون متغيراً أما فيما يخص الطرف الأيمن فيمكن أن يكون تعابير أو قيمة ثابتة أو متغيرا ، أما

إشارة التخصيص (=) فتعني إيجاد القيمة النهائية للطرف الأيمن وحفظها في المتغير الموجود في الطرف الأيسر .

مثال (3-3-1)

خذ بعض الأمثلة على جملة التخصيص المقبولة .

قيمة ثابتة........................ pi = 3.141519 ;

متغيرة .... a = c ;

تعابير أولية...................... h = r+2.3*w ;

تعابير معقدة......... n = (q*(b*k+m)+5)/f ;

في حين الأمثلة التالية غير مقبولة وذلك للسبب المدون قرين كل منها .

الطرف الأيسر عبارة عن تعبير	j+j = 55.5 ;
لا يسمح بإشارة (-) في الطرف الأيسر	-m = 300 ;
الأقواس غير كاملة في التعبير	x1 = -b+(b*b-4.0*a*b))/(2*a) ;

مثال (3-3-2)

الجملة الآتية

```
sum = sum+1 ;
```

تعني إضافة القيمة واحد إلى قيمة المتغير sum بالطرف الأيمن ثم اسناد هذه القيمة إلى المتغير sum بالطرف الأيسر ، أي إذا كانت قيمة sum قبل الإضافة هي 10 ، عندها تصبح قيمة sum تساوي 11 بعد الإضافة .

مثال (3-3-3)

البرنامج الآتي يبين جمل التخصيص المتنوعة مع مخرجاتها .

```
#include <stdio.h>
main()
{
    float a, b, c;
    int d;
    a = 2.9;
    b = 7.7;
    c = a+b;
    d = a<b;
    printf("\n A=%f   B=%f   C==>%f", a, b, c);
    printf("\n a<b ==>%d", d);
}
```

نتيجة تنفيذ هذا البرنامج هي الآتية :-

```
A=2.900000  B=7.700000  C ==>10.600000
a<b ==>1
```

أي أُسْنِدت القيمتان 7.7, 2.9 للمتغيرين b, a على التوالي وأُسْنِدَ مجموعهما للمتغير c ، تلا ذلك مقارنة قيمتي b, a وحيث إن قيمة a أصغر من قيمة b لهذا أسندت القيمة 1 للمتغير d .

مثال (3-3-4)

يمكن إعادة كتابة نفس البرنامج بالمثال السابق باستخدام علامة التخصيص (=) في أكثر من موقع في التعبير نفسه بحيث يكون الناتج مشابها للسابق .

```
#include <stdio.h>
main()
{
    float a, b, c;
    int d;
    c = (a=2.9) + (b=7.7);
    d = (a=2.9) < (b=7.7);
    printf("\n A=%f   B=%f   C ==>%f", a, b, c);
    printf("\n a<b ==>%d", d);
}
```

2) جملة التخصيص الممتدة *Extended Assignment Statement*

المترجم (Compiler) في لغة C يسمح بإستخدام جملة التخصيص الممتدة وهي نادرا ما تجدها في لغات الحاسب الأخرى ومهمتها تخصيص قيمة معينة لاكثر من متغير في نفس الوقت وتخزينها في الذاكرة تحت أسماء تلك المتغيرات.

مثال (3-3-5)

الجمل

```
int x,y,z;
x = 345;
y = 345;
z = 345;
```

تم استخدام ثلاث جمل لإسناد قيمة واحدة 345 للمتغيرات الصحيحة z, y, x وباستخدام جملة التخصيص الممتدة يمكننا إسناد نفس القيمة إلى نفس المتغيرات على النحو الآتي :-

```
x = y = z = 345;
```

بحيث يبدأ التخصيص من اليمين إلى اليسار ، أي تخصيص القيمة 345 للمتغير z أولا ثم تخصيص القيمة المخزنة في المتغير z إلى المتغير y ، وأخيرا تخصيص قيمة y إلى المتغير x .

مثال (3-3-6)

البرنامج الآتي يبين استخدام هذا النوع من جمل التخصيص .

```
#include <stdio.h>
main()
{
    int p, q;
    p = q = (2*5)+7;
    printf("P=%d   Q=%d ", p, q);
}
```

هنا تم تقييم التعبير 7+(2*5) الذي يعطي القيمة 17 وبالتالي إسناد هذه القيمة إلى المتغير q أولا ومن ثم إلى المتغير p ثانيا .

4.3) المؤثرات *Operators*

يوجد في لغة C عدد من المؤثرات هي كالآتي :-

1) المؤثرات الحسابية *Arithmetic Operators*

هناك خمسة مؤثرات حسابية مألوفة في لغة C نوردها بالجدول الآتي :-

المؤثر	معناه
+	الجمع
-	الطرح
*	الضرب
/	القسمه
%	قسمة ينتج عنها الرقم الصحيح الممثل لباقي عملية القسمة

ملاحظة:

يجب أن نأخذ في الاعتبار في حالة القسمة (/) أن يكون الناتج عددا صحيحا في حالة كون عناصر البيانات المستعملة اعدادا صحيحة وكذلك بالنسبة للمؤثر (%) يجب أن تكون عناصر البيانات قيما صحيحة والا فالنتيجة تكون خاطئة .

ويتم تنفيذ التعابير الحسابية بالترتيب الآتي :-

1- الأقواس الداخلي فالداخلي إن وجدت .
2- الضرب (*) والقسمة (/) وباقي القسمة (%) من اليسار إلى اليمين .
3- الجمع (+) والطرح (-) من اليسار إلى اليمين .

مثال (3-4-1)

التعبير

r=p+q/p

وفيه يتم تنفيذ عملية القسمة أولاً ثم عملية الجمع ثانياً

في حين التعبير

x=(p+q)/p

ففيه يتم تنفيذ ما بين الأقواس أي عملية الجمع أولا ثم القسمة ثانيا .

مثال (3-4-2)

لتوضيح العلاقة بين عناصر البيانات من النوع الصحيح والمؤثرات الحسابية ، نورد البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int a, b;
    a = 19;
    b = 5;
    printf("\nA=%d  B=%d  A+B=%d  A-B=%d \n"
         , a, b, a+b, a-b);
    printf("A*B=%d  A/B=%d  A mod B=%d", a*b, a/b, a%b);

}
```

عند تنفيذ هذا البرنامج ستظهر النتائج المشابهة للآتي :-

```
A=19  B=5  A+B=24  A-B=14
A*B=95  A/B=3  A mod B=4
```

مثال (3-4-3)

يمكن استخدام المؤثرات الحسابية مع متغيرات حقيقية وصحيحة.

```
#include <stdio.h>
main()
{
    int a, b;
    float x, y, z;
    a = 100;
    b = 5;
    x = 9.6;
    y = 0.3;
    printf("\n(y+x)/3 ==>%.2f   y-x/3 ==>%.2f\n",(y+x)/3, y-x/3);
    printf("(x*y+x)/3 ==>%.2f\n", (x*y+x)/3);
    z = (x+b) + (a-b)/4;
    printf("((x+b)+(a-b)/4) ==>%.2f", z);
}
```

وقت تنفيذ هذا البرنامج سنحصل على النتائج الآتية:-

```
(y+x)/3 ==>3.30   y-x/3 ==> -2.90
(x*y+x)/3 ==> 4.16
(x+b)+(a-b)/4 ==> 37.60
```

في جملة التخصيص

```
z=(x+b) + (a-b)/4;
```

تم تنفيذ عملية الجمع التي بين القوسين (x+b) أولا ونتج عنها القيمة 14.6 تلا ذلك تنفيذ عملية الطرح التي بين القوسين (a-b) ثانيا ونتج عنها القيمة 95 ثم عملية قسمة 95 على الرقم 4 ثالثا ومنها حصلنا على القيمة الصحيحة 23 وأخيراً تمت عملية جمع القيمتين (14.6+23) لنحصل على القيمة 37.6 التي أسندت للمتغير z .

مثال (3-4-4)

وحتي تكون الصورة واضحة بالنسبة للمؤثرات الحسابية ، وعلى فرض أنه تم الإعلان عن المتغيرات d, c, b, a من النوع الصحيح ، وخصصت لها القيم التالية :-

```
a = 2; b = -5; c = 9; d = -17;
```



عليه يمكن إجراء بعض العمليات الحسابية على مجموعة من التعبيرات التي يوضحها الجدول الآتي :-

التعابير	القيمة
- -c	9
c/a + a	6
c % a	1
a % c	2
-(b+d) % c	4
a % - b*c	18
a % - (b*c)	2
-d % (a*c) % (-d - b)	17
-d /a *c % (a – b)	2
c*d/(a*(-b+a*a-c))	divide error
d*a % ((b+d)/a)*a-d	15

2) المؤثرات العلائقية *Relational Operators*

توجد ستة مؤثرات علائقية نستطيع استخدامها على أي زوج من العناصر يكون ناتجها إما صحيحاً (1) وإما خاطئاً (0) وهي كما يأتي:-

المؤثر	معناه
==	يساوى
!=	لا يساوى
<	أقل من
<=	أقل من أو يساوى
>	أكبر من
>=	أكبر من أو يساوى

مثال (3-4-5)

افرض أن لدينا متغيرين i ,j من النوع الصحيح وحددنا لهما القيمتين 3 ,5- على التوالي ، والمتغير x من النوع الحرفي وخصصنا له الحرف 'w' ، وأجرينا مجموعة من المؤثرات العلائقية على هذه المتغيرات فإن نتائجها ستتضح من خلال الجدول الآتي :-

القيمة	التعبير
1	i != j
1	j*2 == - (i*i+1)
1	'a' -1<x
1	i<j<2
1	!(5+j)
0	x == 'a'
1	'x' == x+1
0	i-j<i+j
0	!x
1	i-3 == j+5
1	j<i>i+j

ملاحظة:

في التعبير i < j < 2 تمت المقارنة بين قيمة المتغير i والمتغير j والنتيجة خطأ أي القيمة 0، تلا ذلك مقارنة هذه القيمة مع العدد 2 فتكون النتيجة صحيحا أي القيمة 1 ، أما التعبير i-j < i+j فيعطي القيمة 0 لأن 8 ليست أصغر من 2- ، كذلك بالنسبة للتعبير x == 'x'+1 فهو يعني أضف 1 إلى قيمة x وهي w سينتج عنها الحرف الآتي للحرف w وهو x وبالتالي فإن التعبير يصبح x == x وينتج عنه القيمة 1 .

3) المؤثرات المنطقية *Logical* **Operators**

هذه المؤثرات ينتج عنها إما قيمة صحيح (1) أو قيمة خطأ (0) وهذه المؤثرات هي :-

المؤثر	معناه
&&	مؤثر ثنائي ، يكون التعبير صحيحا إذا كان كل من العنصرين صحيحا .
||	مؤثر ثنائي ، يكون التعبير صحيحا إذا كان أحد العنصرين أو كلاهما صحيحا
!	مؤثر آحادي ، نفي العنصر أو النقيض

مثال (3-4-6)

التعبير

```
( a > 0 ) && ( a <= b )
```

تكون نتيجته صحيحا إذا كانت a أكبر من صفر وأقل من b أو مساوية له وتكون خاطئا إذا كانت a أقل من صفر أو أكبر من b .

المؤثر (!) يعكس قيمة التعبير المنطقي ، فمثلاً:

```
( ! found)
```

يكون خاطئا إذا كانت قيمة found صحيحا .

مثال (3-4-7)

باستخدام المؤثرات المنطقية ، المطلوب كتابة برنامج كامل مهمته الحصول على جدول الصدق لمتغيرين .

```
#include <stdio.h>
main()
{
    int i, j;
    i =1;
    j = 1;
    printf("---------------------- \n");
```

```
    printf("  I  J  I&&J  I||J  !I\n");
    printf("------------------------ \n");
    printf(" %d   %d   %d   %d   %d\n",
        i, j, i&&j, i||j, !i);
    i = 1; j = 0;
    printf(" %d   %d   %d   %d   %d\n",
        i, j, i&&j, i||j, !i);
    i = 0; j = 1;
    printf(" %d   %d   %d   %d   %d\n",
        i, j, i&&j, i||j, !i);
    i = 0; j = 0;
    printf(" %d   %d   %d   %d   %d\n",
        i, j, i&&j, i||j, !i);
    printf("------------------------ \n");
}
```

الناتج وقت تنفيذ هذا البرنامج يكون كالجدول المشابه للآتي :-

```
-------------------------
I   J  I&&J  I||J  !I
-------------------------
1   1   1    1    0
1   0   0    1    0
0   1   0    1    1
0   0   0    0    1
-------------------------
```

4) مؤثرات الزيادة والنقصان *Increment And Decrement Operators*

تتميز لغة C ببعض المؤثرات التي قد لا تجدها في بعض اللغات الاخرى وهي مؤثرات الزيادة والنقصان التي لها الشكل العام الآتي :-

```
op variable
variable op
```

حيث op مؤثر الزيادة (++) ومؤثر النقصان (--) الذي يمكن استعماله مع المتغيرات فقط ووضعه قبل المتغير أو بعده .

فالتعبير m++ يعني أضف 1 إلى قيمة m قبل احتساب التعبير ، على حين التعبير m++ يعني استخدم القيمة الحالية للمتغير m في التعبير ثم أضف 1 إليه

أى عند استعمال m++ نزيد قيمة m قبل احتساب التخصيص وعند استعمال m++ يحسب التخصيص قبل زيادة m .

وينطبق هذا أيضاً في حالة مؤثر النقصان (--) .

مثال (8-4-3)

المطلوب إيجاد ناتج البرنامج الآتي مع الشرح .

```
#include <stdio.h>
main()
{
    int i, j, k;
    i = j = 2;
    k = ++i;
    printf("I==>%d  K==>%d\n", i, k);
    k = j++;
    printf("J==>%d  K==>%d", j, k);
}
```

الناتج هو الآتي :-

```
I==>3   K==>3
J==>3  K==>2
```

في بداية البرنامج خصصت القيمة 2 للمتغيرين j, i وفي جملة التخصيص ;k = ++i أُضيف 1 للمتغير i أولا فأصبحت قيمته 3 ثم خصصت هذه القيمة للمتغير k ثانيا وبالتالي ينتج العدد 3 لكل من المتغيرين k, i .

وفي الجملة الثانية ;k = j++ خصصت قيمة المتغير j وهي العدد 2 للمتغير k أولا ثم أُضيف 1 قبل الطباعة للمتغير j فنتج عنه العدد 3 .

مثال (3-4-9)

انظر إلى البرنامج الآتي

```
#include <stdio.h>
main( )
{
    int s1, s2, a = 5, b = 3;
    s1 = a+ ++a;  /* Increment then add */
    s2 = b+ b++;  /* Add then Increment */
    printf("S1 ==> %d   S2 ==> %d\n", s1, s2);
}
```

عند تنفيذه ينتج عنه الآتي :-

S1 ==> 12   S2 ==> 6

بالأمر الأول زيادة واحد للمتغير a فأصبحت تساوي 6 أولا ثم جآءت عملية الجمع ونتج عنها القيمة 12 التي خصصت للمتغير s1 ثانيا ، أما فيما يخص الأمر الثاني فقد جآءت فيه عملية الجمع اولا ونتج عنها القيمة 6 التي أسندت للمتغير s2 ثم زيادة واحد للمتغير b ثانيا .

مثال (3-4-10)

افرض أن المتغيرات c, b, a من النوع الصحيح وخُصِّصت القيمة 3 لكل منها ، فيما يأتي مجموعة من التعبيرات مقرونة بالنتائج أمام كل منها :-

القيمة	التعبير
3	a++
4	++b
3	c--
-3	- - -a
-4	- ++a
1	a- --b
6	b++ + c++
8	++b + ++c
7	++b + c++
6	b-- + c--
8	++c + c
3	-a + --b + ++c
9	a++ /2 + --b* ++c
6	--c* ++b - --a

5) المؤثرات المركبة *Compound Operators*

ميزة أخرى تضاف إلى لغة C وهي استخدام المؤثرات الحسابية المعروفة (+ , - , * , / , %) والمؤثرات الخاصة بالبت BIT وهي (| , & , << , >>) مع إشارة التخصيص (=) تحت اسم المؤثرات المركبة والتي تعتبر طريقة مختصرة لجملة التخصيص التقليدية .

الشكل العام :

variable op= expression

حيث op يمكن أن تكون إحدى المؤثرات الحسابية أو المؤثرات الخاصة بالبت . ولهذه المؤثرات نفس الأسبقية وتنفيذها يتم من اليمين إلى الشمال .

مثال (3-4-11)

خذ مثلا التعبير x += 9;

مكافئ للتعبير x = x+9;

وفيهما يعني أضف 9 للمتغير x بالطرف الأيمن ثم خصص هذه القيمة للمتغير x الموجود في الطرف الأيسر.

مثال (3-4-12)

الجدول الآتي يبين الفرق بين جملة التخصيص التقليدية والمركبة .

التعبير باستخدام جملة	
التخصيص التقليدية	التخصيص المركبة
a = a+b;	a += b;
a = a-b;	a -= b;
a = a*b;	a *= b;
a = a/b;	a /= b;
a = a%b;	a %= b;

مثال (3-4-13)

المطلوب شرح البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int p, q, r;
    p = 2;
    q = 3;
    r = 10;
    r /= p+q;
    printf("r /= p+q ==>%d", r);
}
```

البرنامج وقت تنفيذه يطبع السطر الآتي :-

```
r /= p+q ==>2
```

حيث نفذ التعبير p+q وأعطي القيمة 5 ثم قسمت قيمة r وهي 10 على القيمة 5 فنتج عنها القيمة 2 التي أسندت للمتغير r أي أن هذا التعبير مكافئ للتعبير

```
r = r/(p+g);
```

وليس للتعبير

```
r = r/p+q;
```

الذي فيه قسمت قيمة r وهي 10 على قيمة p وهي 2 فكان الناتج 5 ثم أضيفت إلى قيمة q وهي 3 فأصبح الناتج هو 8 .

وعلى نفس الطريقة التعبير

```
i *= j-k;
```

مكافيء للتعبير

```
i = i*(j-k);
```

وليس للتعبير

```
i = i*j-k;
```

والتعبير

```
a %= b*c/c;
```

مكافئاً للتعبير

```
a = a%(b*c/c);
```

5.3) أولويات تنفيذ المؤثرات *Operators Precedence*

كلمة (Hierarchy) تعني الأولوية في تنفيذ التعبير وتتضح الحاجة الماسة إلى هذا في حالة وجود تعبير به عدد من المؤثرات المختلفة ، وإذا أردنا أن يأخذ التعبير مسارا محددا لتقييمه يجب استخدام الاقواس وذلك حسب ما يقتضيه التعبير .

يتم تنفيذ المؤثرات في لغة C وفق سلم الأولويات التالية من أعلى إلى أسفل.

المؤثر	التنفيذ
( ) [ ] -> .	من اليسار إلى اليمين
! ++ -- -(unary)	من اليمين إلى اليسار
* / %	من اليسار إلى اليمين
+ -	من اليسار إلى اليمين
< <= > >=	من اليسار إلى اليمين
!= ==	من اليسار إلى اليمين
&&	من اليسار إلى اليمين
||	من اليسار إلى اليمين
= += -= *= /= %=	من اليمين إلى اليسار

6.3) تحويل النوع *Type Conversions*

في كثير من الأحيان يكون التحويل من قيم صحيحة إلى قيم حقيقية أو بالعكس مطلوبا وعليه تتم عملية التحويل بالصورة الآتية :-

(type) expression

مثال (3-6-1)

البرنامج الآتي يوضح عملية التغيير :

```
#include <stdio.h>
main( )
{
    int a =7, b =2;
    float a = 3.9, b = 5.4;
    printf("(float) %d/%d ==> %.2f", a, b, (float) a/b);
    printf("int(%.1f) + int(%.1f) ==> %d", a, b, (int) a + (int) b);
}
```

وعند تنفيذه ينتج عنه الآتي :-

```
(float) 7/2 ==> 3.50
int(3.9) + int(5.4) ==> 8
```

وفيه تغيرت القيمة الناتجة من عملية القسمة من النوع الصحيح إلى الحقيقي بالسطر الأول ، أما بالسطر الثاني فتحولت القيم الحقيقية إلى قيم صحيحة .

أيضاً يتم التحويل عندما يربط المؤثر عنصرين من نوعين مختلفين عندها يجب تحويلهما إلى نوع عام أو موحد لأي مؤثر في التعبير ، ويتم ذلك بالآتي:-

1) إذا كان العنصر من نوع char أو short يتم تحويله إلى أعداد صحيحة int .
2) القيم الحقيقية float إلى قيم حقيقية مضاعفة double .
3) إذا كان أحد العناصر حقيقيا مضاعفا double فإن باقي العناصر يتم تحويلها إلى عدد مضاعف والنتيجة من النوع المضاعف double .
4) إذا كان احد العناصر من النوع long فإن باقي العناصر تحول إلى نوع طويل long والنتيجة من النوع الطويل long .
5) إذا كان أحد العناصر طبيعيا بدون إشارة unsigned فإن باقي العناصر يتم تحويلها إلى طبيعي ، والنتيجة من النوع الطبيعي unsigned .
6) وأخيراً فإن عنصرى المؤثر يجب أن يكونا من نوع int وكذلك النتيجة

مثال (3-6-2)

اكتب برنامجا لحساب قيمة المعادلة الآتية :-

result = d-x+(i*c)

حيث :

d = 0.2E02 , x = 2.5 , c = '#' , i = 10

مع طباعة القيم الداخلة والنتيجة النهائية للمعادلة .

```
#include <stdio.h>
main()
{
    int i; float x;
    double d, RESULT;
    char c = '#';
    i = 10; x = 2.5;
    d = 0.2E02;
    RESULT = d-x+(c*i);
    printf("\nRESULT ===>%.3E When :\n", RESULT);
    printf("I=%d, X=%f", i, x);
    printf(", D=%.2E, and C='%c'", d, c);
}
```

في بداية البرنامج تم الإعلان عن المتغيرات المعطاة حسب نوعية قيمة كل منها تلا ذلك حساب قيمة المعادلة والنتائج المشابهة للآتي وقت تنفيذ البرنامج .

```
RESULT =3.675E+02 When :
I=10, X=2.500000, D=2.00E+01, and C='#'
```

نلاحظ أن المعادلة

```
RESULT = d-x+(c*i);
```

جاءت من النوع المضاعف double والناتجة من الخطوات الآتية :-

1) تمت العملية التي بداخل الأقواس (c*i) أولاً والنتيجة هي 350 حيث الرمز (#) يساوي 35 انظر ملحق (3) .
2) تحويل قيمة العنصر x من حقيقي float إلى مضاعف double حيث أصبحت 0.025E02
3) تأتي عملية الطرح d-x حيث ينتج عنها القيمة 175E02
4) تحول القيمة الصحيحة 350 إلى قيمة مضاعفة double فتصبح 3.5E02
5) أخيرا تأتي عملية جمع الخطوتين (3 ,4) لتعطي النتيجة النهائية للمعادلة وهي 3.675E+02

7.3) تمرينات Exercises

1) على فرض أن لدينا الإعلان

```
int i = 5, j = -3;
char  x = 'A';
```

أوجد ناتج الآتي :-

```
a) printf("%d ", i*(-j));
b) printf("%o ", i*(-j));
c) printf("%d ", x+2 == 67);
e) printf("%d ", ++j + -- i);
f) printf("%d ", i++ + j++ + ++i );
g) printf("%d ", (i != 5) || ( !( i<j )) );
h) printf("%d ", !(i == j) && i+j != i*(i-j));
i) printf("%d ", (i == j) || i+j == i*(i-j));
k) printf("%d ", (i <= j) && ( j != i-8) || ( j<0 ) );
```

2) أي من جمل التخصيص الآتية مقبولة ؟

```
a)  A B = a + b + c , 2d;
b)  5m=5[m + 3n] ;
c)  X = Y = 5 = Z ;
d)  char = c1 - c2 + 'a' ;
e)  ++k / = k++ ;
```

3) أوصف ناتج الآتي :-

```
a)  printf("I=%05d ",555);
b)  printf("k -->%5s","abc");
c)  printf("k -->%E",23.689);
d)  printf("k -->%.2f", 23.689);
e)  printf("k -->%.0f",23.689);
f)  printf("k -->%d",0<! (-9));
```

4) المطلوب قراءة مجموعة من البيانات الصحيحة من السطر الأول ، والحقيقية من السطر الثاني، والحرفية من السطر ثالث مع طباعة البيانات الحرفية في السطر الأول والحقيقية في الثاني والصحيحة في السطر الثالث.

7.3) تمرينات Exercises

1) على فرض أن لدينا الإعلان

```
int i = 5, j = -3;
```

أوجد ناتج الآتي :-

```
a) printf("A=%d ", i/=3);
b) printf("B=%d ", (i*(j*j)+i*4)) <= 'A' );
c) printf("C1=%d C2=%d ", ++i , ++i);
d) printf("D=%d ", i-- *i);
e) printf("E=%d ", i++ + j++ + ++i );
f) printf("F=%d ", (i != 5) || ( !( i<j )) );
g) printf("G=%d ", !(i == j) && i+j != i*(i-j));
h)  printf("H=%d ", (i == j) || i+j == i*(i-j));
i) printf("I=%d ", (i <= j) && ( j != i-8) || ( j<0 ) );
```

2) أوجد قيمة المتغير R لكل تعبير من التعابير الآتية في حالة :-

```
int R,a=4, b=9 , c=7;

a)  R=( b%a + ++a/2/2 ) % 2
b)  R=b*4%(b-a) <= 1
c)  R=++a + b-- ==  (a * (a-2))
d)  R=a+a%b/ (b/5-1)
e)  R=c++ % a + c/3
f)  R=b+b/a-(c+ ++a)/ 2
g)  R=a+(++b*a) %c*c-a
```

3) قم بكتابة برنامجاً كاملاً يقرأ نصف قطر دائرة R ثم يحسب ويطبع مساحتها A ومحيطها B علما بأن :

$$A = \pi R^2$$
$$B = 2\pi R$$

4) المطلوب قراءة مجموعة من البيانات الصحيحة من السطر الأول ، والحقيقية من السطر الثاني، والحرفية من السطر ثالث مع طباعة البيانات



```
#include <stdio.h>
main()
{
      short a, b, c = 3 , x = 4;
      a = b = ++x;     b = --a - c++ + 2;
      printf("\n A=%d  B=%d  C=%d ", a, b, c);
      a = b = x - (++x - x);
      b /= a + b - x;
      c = (x / 2 + x) / 2;
      printf("\n A=%d  B=%d  C=%d ",a, b, c);
      int y = 7, z = 2; float w;
      w = (float) y/z + (float) ++ y/--z +y%z;
      printf ("\\n y=%d  z=%d  w= %. 2f", y,z,w);
```

9) صمم برنامجاً يقرأ عدد أيام ثم تحويله إلى أسابيع وأيام مع الطباعة الناتج فمثلا 27 يوم يطبع عدد الأسابيع 3 وعدد الأيام المتبقية 6 .

10) المطلوب كتابة برنامج لإيجاد قيمة المتغير y حيث :

$$y=\left[\frac{y-b}{5}\right]^{n}+\frac{b}{3}$$

11) اكتب برنامجا لإسناد اسم الطالب SOFYAN SOHIAB GAYED إلى المتغير name وعمره 4.5 إلى age ثم أكتب دالة الإخراج المناسبة لإظهار هذه البيانات بالشكل التالي :-

Hi **** SOFYAN SOHIAB GAYED **** your age is 4.5 year old

12) اكتب برنامجا كاملا لقراءة العجلة الثابتة A والزمن T ثم احسب المسافة D والسرعة النهائية V حيث :

$$D=\frac{1}{2}AT^2$$

$$V=AT$$

4

الفصل الرابع

الاختيارات والتبديل

تناولنا في الفصول السابقة بعض البرامج التي تتكون من عدد من الجمل المتسلسلة بدون أي اختيار ، وحيث إن عملية الاختيار تكون واردة في بعض المسائل ، عليه يمكن تقديمها الآن .

1.4) جملة إذا if *Statement*

تستخدم هذه الجملة لاتخاذ القرار حيث تقوم بتقييم التعبير المنطقي الذي يحدد نتيجة القرار الصحيح (true) أو الخطأ (false) وهو ما يسمى بالاختيار ذي التفريعين (two way choice) .

تستخدم جملة (if) بالصورة الآتية :-

```
if(condition)
   statement_1
next statement
```

في هذه الحالة يتم تنفيذ جملة واحدة statement_1 إذا كانت نتيجة الشرط condition التي يجب وضعها ضمن القوسين () صحيحة يلي ذلك تنفذ الجملة التالية next statement ، أما إذا كانت نتيجة الشرط خاطئة فلا تنفذ الجملة statement_1 بل تنفذ الجملة التالية next statement أي ان الجملة التالية تنفذ في كلا الحالتين .

مثال (4-1-1)

انظر إلى هذا الإيعاز

```
if ( grade >= 85 )
    printf("CONGRATULATIONS !");
printf("YOUR GRADE IS %.2f", grade);
```

فهو يتكون من :

1) الشرط (grade >= 85) أي هل الدرجة grade أكبر من أو تساوي 85 .
2) وطباعة كلمة التهنئة ! CONGRATULATIONS والدرجة إذا كـــان شرط if تحقق .
3) وطباعة الدرجة فقط إذا لم يتحقق الشرط .

مثال (4-1-2)

إذا اعطيت البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int  number;
    printf("\nEnter number ==> ");
    scanf("%d", &number);
    if( number > 0 )
      printf("The %d is positive number", number);
    printf("\nAll done");
}
```

ما هو الناتج في الحالتين :-

1) إدخال القيمة 500
2) إدخال القيمة 25-

الجواب : عند التنفيذ وإدخال القيمة بالحالة (1) على الصورة

```
Enter number ==>500
```

سيطبع البرنامج

```
The 500 is positive number
All done
```

لأن الشرط number>0 تحقق وبالتالي تم تنفيذ الجملة الموالية لـــ if يتبعها تنفيذ الجملة التالية وهي طباعة All done .

أما في حالة إدخال القيمة كلآتي :-

```
Enter number ==> -25
```

هنا سيطبع الجملة (الكل انتهى)

```
All done
```

لأن الشرط لم يتحقق وبالتالي تم تجاهل جملة الطباعة الاولى .

2.4) الجملة المركبة Compound Statement

الجملة المركبة هي التي تحتوي على تسلسل من جملتين متتاليتين أو أكثر محاطة بقوسي الفئة { } .

مثال (4-2-1)

خذ مثلا جملة if الآتية :_

```
if(condition)
{
    statement_1;
    statement_2;
    . . .
    statement_n;
}
```

وفيها إذا تحقق الشرط condition عندها تنفذ كل الجمل المحاطة بين قوسي الفئة ، أما إذا كان الشرط خطأ فإنه يتم تجاهل هذه الجمل جميعها وينفذ السطر الذي يلي هذه الجمل .

مثال (4-2-2)

فيما يلي بعض الأمثلة توضح استخدام الجملة المركبة

(1)

```
{
    counter += 1;
    sum += number;
}
```

(2)

```
if( wages >= 0 )
{
    float  hours,rate;
    scanf("%f   %f", &hours, &rate);
    wages = hours*rate;
    printf("%15s  %.3f", name, wages);
    total += wages;
}
```

مثال (4-2-3)

المطلوب كتابة برنامج لاستقبال عددين مع طباعتهما تصاعديا .

```
#include <stdio.h>
main( )
{
    float num1, num2, temp;
    printf("Enter 2 float numbers ==> ");
    scanf("%f  %f", &num1, &num2);
    if( num1 >= num2 )
    {
            temp = num1;
            num1 = num2;
            num2 = temp;
    }
    printf("num1=%.2f   num2=%.2f", num1, num2);
}
```

في هذا البرنامج وعند تنفيذه سيظهر السطر

```
Enter 2 float numbers ==>
```

على شاشة العرض ، وإذا تم إدخال القيمتين 4.0 , 9.0 وتخصيصهما للمتغيرين num2, num1 على التوالي ، ومن هنا فالشرط (num1>=num2) لم يتحقق والسبب أن قيمة المتغير num1 لم تكن أكبر أو تساوي قيمة المتغير num2، لذلك لم تنفذ الجملة المركبة والموالية لجملة if بل انتقل التنفيذ لجملة الطباعة مباشرة وبالتالي يظهر الناتج الآتي :-

num1=4.00   num2=9.00

وإذا ما نُفذ البرنامج مرة أخرى وجرى إدخال القيمتين 9.0 , 4.0 واسندت القيمة 9 للمتغير num1 والقيمة 4 للمتغير num2 فسيأتي بعدها عملية المقارنة والتي ينتج عنها تحقيق الشرط (num1>=num2) وبالتالي يجرى تنفيذ الجملة المركبة

```
{
    temp = num1;
    num1 = num2;
    num2 = temp;
}
```

التي تعني إجراء عملية تبديل هاتين القيمتين بالطريقة الآتية :-

		num1	num2	temp
قيم المتغيرات المبدئية		9.0	4.0	
قيم المتغيرات بعد الجملة	temp=num1;	9.0	4.0	9.0
قيم المتغيرات بعد الجملة	num1=num2;	4.0	4.0	9.0
قيم المتغيرات بعد الجملة	num2=temp;	4.0	9.0	9.0

بعده تأتي جملة الطباعة وينتج عنها

num1=4.00   num2=9.00

مثال (4-2-4)

البرنامج الآتي يوضح كيفية الإعلان عن عدد من المتغيرات باسم معرف واحد .

```
#include <stdio.h>
main( )
{
    /* main Block */
    int x = 555;
        /* Block one */
```

```
    {
        float x = 12.34;
        printf("X in block one ==> %.2f\n", x);
    }
        /*  Block two  */
    {
        char x[20] = "Welcome user";
        printf("X in block two ==> %s\n", x);
    }
    printf("X in main block ==> %d\n", x);
}
```

تنفيذ هذا البرنامج سينتج عنه الآتي :-

```
X in block one ==> 12.34
X in block two ==>Welcome user
X in main block ==> 555
```

وفيه الإعلان عن متغيرات مختلفة تحت اسم معرف واحد داخل عدد من الجمل المركبة أو ما يعرف بالقالب (Block) وهي على النحو الآتي :-

1) في بداية القالب الرئيسي (Main Block) تم استخدام المعرف x كمتغير من النوع الصحيح مع إسناد القيمة 555 له وبذلك يعتبر متغيرا خارجيا بالنسبة للقوالب الأخرى .
2) في بداية القالب الأول تم الإعلان عن المعرف x كمتغير من النوع الحقيقي مع إسناد القيمة 12.34 له وطباعة قيمته وبذلك يعتبر x متغيرا محليا لهذا القالب .
3) ثم جاء القالب الثاني حيث أعلن عن نفس المعرف x كمتغير من نوع السلسلة الحرفية وإسناد العبارة Welcome user لهذا المتغير مع طباعة قيمته .
4) أخيراً طباعة قيمة المتغير x المخصصة اليه بالقالب الرئيسي .

3.4) جملة إذا ... وإلا if-else Statement

تسمي هذه الجملة بالجملة الكاملة وتعني أنه إذا تحقق شـرط if فـافعل كـذا وإلا فافعل كذا .

الصيغة العامة

```
if( condition )
    statement_1;
else
    statement_2;
```

لو تحقق الشرط condition عندها نفذ الجملـة statement_1 وإذا لـم يتحقـق الشرط فنفذ الجملة statement_2

مثال (4-3-1)

جملة if الآتية :-

```
if (grade >= 50)
    printf("PASS %.2f", grade);
else
    printf("FAIL %.2f", grade);
```

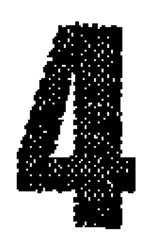
تتكون من ثلاث عبارات :-

1) الشرط (grade >= 50)
2) طباعة كلمة PASS مع قيمة المتغير grade إذا تحقق الشرط .
3) طباعة كلمة FAIL مع قيمة المتغير grade إذا لم يتحقق الشرط .

مثال (4-3-2)

انظر إلى الايعاز الآتي :-

```
if(grade<50)
    ;    /* empty statement   */
else
    printf("\nGrade grather than or equal 50");
```



وفيه استُخْدِمَت الفاصلة المنقوطة (;) بعد الشرط مباشرة وهنا تعني إذا ما تحقق هذا الشرط أي الدرجة أقل من 50 فلا تنفذ أي شئ ، أما إذا كان غير ذلك فنفذ جملة الطباعة وهي الدرجة أكبر من أو تساوي 50 .

مثال (4-3-3)

المطلوب كتابة برنامج لقراءة عدد صحيح وطباعة الرسالة positive number إذا كان موجبا أو الرسالة zero or negative إذا كان غير ذلك.

```
#include <stdio.h>
main()
{
    int number;
    printf("\nType your number ==> ");
    scanf("%d", &number);
    if( number > 0 )
      printf("The %d is positive number", number);
    else
      printf("The %d is zero or negative number", number);
    printf("\nAll done");
}
```

عند تنفيذ هذا البرنامج وإدخال رقم سالب وليكن 77- كالآتي :-

```
Type your number ==> -77
```

عندها لا يتحقق شرط جملة if وعليه سيتم تنفيذ الجملة الموالية لـ else وهي جملة الطباعة الأولى ثم تنفذ الجملة الثانية والموالية لهذه الجملة ويكون الناتج :-

```
The -77 is zero or negative number
All done
```

أما في حالة إدخال قيمة موجبة ولتكن 75 فهنا يتم تنفيذ جملة الطباعة الأولى لأن الشرط تحقق ويتغاضي المترجم عن جملة الطباعة الثانية التي تلي else ويقفز إلى جملة الطباعة الأخيرة ويكون الناتج :-

```
The 75 is positive number
All done
```

4.4) جملة إذا المتداخلة *Nested* if *Statement*

هذا النوع من الجمل يتألف من مجموعة if مع أي عدد من if-else المتداخلة، عليه يجب إجراء هذا النوع من التداخل بدقة حتى لا يكون هناك أي أخطاء وقت الاستعمال .

الشكل العام

```
if( condition_1 )
{
    statement_11;
    statement_12;
}
else
    if( condition_2 )
    {
        statement_21;
        statement_22;
     }
     else
        if( condition_3 )
statement_31;
```

وهي تعني إذا كان الشرط condition صحيحا تنفذ الجملة أو مجموعة الجمل الواقعة ما بين if وأول else if ويتم تجاهل بقية الجمل ، اما إذا كان الشرط خطأ فإنه يتم الانتقال إلى أول شرط else if وهكذا .

وعليه وحتي يكون هذا النوع من الجمل سهل المتابعة والفهم وعدم الارتباك عند تتبعه ينبغي أن تأتي كل else تحت كلمة if التي تتبعها كما هو موضح في الشكل العام .

مثال (4-4-1)

المطلوب إعادة كتابة البرنامج بالمثال (4-3-3) بحيث تطبع الرسالة المناسبة في حالة ما إذا كان العدد المدخل موجبا أو سالبا أو صفرا .

لكتابة هذا البرنامج يجب استخدام أكثر من جملة if وذلك لمقارنة العدد مع الصفر ، وعليه قد يكون أكبر من الصفر أو أصغر من الصفر وإلا فإن العدد يكون صفرا .

```
#include <stdio.h>
main()
{
    int number;
    printf("\nEnter your number ==> ");
    scanf("%d", &number);
    if( number > 0 )
      printf("The %d is positive number", number);
    else
      if( number < 0 )
        printf("The %d is negative number", number);
      else
        printf("The %d is zero number", number);
    printf("\nAll done");
}
```

هنا تم استخدام جملتي if مع else للحصول على المطلوب ، وإذا ما نفذ هذا البرنامج في ثلاث حالات فسيكون الناتج كالآتي :-

1) في حالة إدخال القيمة السالبة 1234-

```
Enter your number ==> -1234
The -1234 is negative number
All done
```

2) في حالة إدخال القيمة 0

```
Enter your number ==> 0
The 0 is zero number
All done
```

3) في حالة إدخال القيمة 876

```
Enter your number ==> 876
The 876 is positive number
All done
```

لاحظ أن جملة All done تم طباعتها في كل حالة من الحالات الثلاث.

مثال (4-4-2) المطلوب اختصار جمل if الآتية : -

```
if( grade > 70 )
    printf("Excellant Your Grade is %.2f", grade);
if ( grade < 30 )
    printf("Dismal Your Grade is %.2f", grade);
if( ( grade <= 70 ) && (grade>=30) )
    printf("Typical Your Grade is %.2f", grade);
```

يمكن أن يتم الاختصار باستعمال جملة if-else كما يلي :-

```
if( grade > 70 )
    printf("Excellant Your Grade is %.2f", grade);
else
     if( grade < 30 )
       printf("Dismal Your Grade is %.2f", grade);
     else
        printf("Typical Your Grade is %.2f", grade);
```

مثال (4-4-3)

اكتب برنامجا لقراءة أي رمز معين (Character) مع عمل الآتي :-

1) إذا كان الرمز المدخل هو (?) عندها اطبع الرسالة الآتية :-

```
Goodbye no next character
```

2) إذا كان الرمز حرفا أو رقما فيجب أن يحل محله الرمز الآتي له ، فمثلا الحرف X يحل محله Y فيما عدا : إذا كان الرمز 9 يحل محله

الرمز 0 ، إذا كان الرمز z يحل محله الرمز a ، إذا كان الرمز Z يحل محله الرمز A .

3) إذا كان الرمز غير ما ذكر أعلاه فإنه يحل محله الرمز * .

لكتابة هذا النوع من البرامج وجب استخدام أكثر من جملة if-else وذلك لكثرة المقارنات المستعملة في هذا المثال .

```
#include <stdio.h>
main()
{
    char  char_in, char_out;
    printf("\nEnter a character ==> ");
    scanf("%c", &char_in);
    if( char_in != '?' )
    {
      if( (char_in >='0') && (char_in <'9')
       || (char_in >='A') && (char_in <'Z')
       || (char_in >='a') && (char_in <'z') )
       char_out = char_in+1;
      else
        if(char_in == '9')
          char_out = '0';
        else
          if(char_in == 'z')
                char_out = 'a';
          else
                if(char_in == 'Z')
                  char_out = 'A';
                else
                  char_out = '*';
      printf("The input character is ===> '%c' ", char_in);
      printf("and next character is ===> '%c' \n", char_out);
    }
    printf("Goodbye no next character.");
}
```

وللتأكد من صحة البرنامج يمكن تنفيذه أكثر من مرة وذلك بإدخال عدد من الرموز المختلفة كما يلي :-

```
Enter a character ==> z
```

```
The input character is ==> 'z' and next character is ==> 'a'

Enter a character ==> H
The input character is ==> 'H' and next character is ==> 'I'

Enter a character ==> 6
The input character is ==> '6' and next character is ==> '7'

Enter a character ==> ?
Goodbye no next character.
```

5.4) جملة التبديل *Switch Statement*

كما لاحظنا سابقا في جملة if فإن المقارنة تتم بين قيمتين وتكون النتيجة إما صحيحة أو خاطئة ولكن في بعض الأحيان علينا أن نقارن بين عدد من الحالات تبعا لشروط مختلفة .

في لغة C هناك أمر switch الذي مهمته القيام بعدة مقارنات عوضا عن استخدام جملة if .

الشكل العام هو :

```
switch (expression)
{
   case constant1 :  statement (s);
                     break;
   case constant2 :  statement (s);
                     break;
   case constant3 :  statement (s);
                     break;
                     . . .
   default :  statement (s);
}
```

حيث :

* switch جملة التبديل التي تبدأ وتنتهي بالقوسين { } يمكن بها تنفيذ واحد من عدة اختيارات متاحة تبعا لقيمة التعبير expression .

* expression هو ذلك التعبير الذي يجب أن تكون نتيجته من النوع الحرفي char أو النوع الصحيح int أو المنطقي logical .
* case تمثل عنوان الحالة المطلوب تنفيذها بعد احتساب التعبير expression .
* constant قيمة التعبير expression يتبعه الشارحة (:) يتبع ذلك استخدام جملة أو عدة جمل ولا يجب أن تكون constant بترتيب معين.
* break تستعمل عند آخر كل حالة case لتفادي استمرار بقية الحالات الموالية ، وإذا لم تستخدم فإن التنفيذ ينتقل إلى الحالة التالية لهذه الحالة.
* default تعني حالة إسقاط وهي اختيارية وتنفذ عندما تكون قيمة التعبير expression لا تتحقق مع أي حالة من الحالات السابقة.

مثال (4-5-1)

اكتب برنامجا لإدخال أحد المؤثرات (+, -, *, /) مع كتابة الرسالة المناسبة التي تدل على نوع المؤثر فمثلا في حالة إدخال المؤثر (+) تطبع الرسالة

THE '+' IS ADDITION OPERATOR

وطباعة الرسالة المناسبة في حالة إدخال أي حرف غير ما ذكر أعلاه .

```
#include <stdio.h>
main()
{
        char op;
        printf("\nEnter operator please ==>");
        scanf("%c", &op);
        switch (op)
        {
          case '+' : printf("THE '%c' IS ", op);
                     printf("ADDITION OPERATOR\n");
                     break;
          case '-' : printf("THE '%c' IS ", op);
```

```
            printf("SUBTRACTION OPERATOR\n");
            break;
    case '*' : printf("THE '%c' IS ", op);
            printf("MULTIPLICATION OPERATOR\n");
            break;
    case '/' : printf("THE '%c' IS ", op);
            printf("DIVISION OPERATOR\n");
            break;
    default : printf("***** ERROR INPUT *****\n");
  }
}
```

عند تنفيذ البرنامج تظهر الرسالة :

Enter operator please ==>

مطالبة بإدخال المؤثر ثم تأتي جملة switch وهي لا تنتهي بفاصلة منقوطة (;) حيث تحتوي على التعبير op من النوع الحرفي char .

فإذا ما تم إدخال المؤثر + فعندها يتم تنفيذ الحالة الأولى ، ومن ثم تطبع الرسالة الآتية :-

THE '+' IS ADDITION OPRETOR

ثم ينتقل التحكم إلى الأمر الآتي وهو

break;

والذي بدوره يحول التحكم إلى القوس المغلق لجملة switch ، وهكذا لباقي الحالات ، أما في حالة إدخال أي حرف من الحروف غير المشار إليها ، فعندها تنفذ جملة الإسقاط default حيث ينتج عنها طبع الرسالة :

***** ERROR INPUT *****

مثال(4-5-2)

المطلوب كتابة البرنامج بالمثال السابق بدون استخدام جملة break مع بعض الحالات .

```
#include <stdio.h>
main()
{
        char op;
        printf("\nEnter operator please ==>");
        scanf("%c", &op);
        switch (op)
        {
          case '+' : printf("THE '%c' IS ", op);
                     printf("ADDITION OPERATOR\n");
          case '-' : printf("THE '%c' IS ", op);
                     printf("SUBTRACTION OPERATOR\n");
                     break;
          case '*' : printf("THE '%c' IS ", op);
                     printf("MULTIPLICATION OPERATOR\n");
          case '/' : printf("THE '%c' IS ", op);
                     printf("DIVISION OPERATOR\n");
          default : printf("***** ERROR INPUT *****\n");
        }

}
```

إذا ما نفذ هذا البرنامج لأكثر من مرة فستكون المدخلات والمخرجات كالآتي :-

```
Enter operator please ==>+
THE '+' IS ADDITION OPERATOR
THE '+' IS SUBTRACTION OPERATOR
Enter operator please ==>*
THE '*' IS MULTIPLICATION OPERATOR
THE '*' IS DIVISION OPERATOR
***** ERROR INPUT *****
```

في هذا البرنامج لو تم إدخال المؤثر + فالنتيجة ستكون مشابهة للآتي :-

```
THE '+' IS ADDITION OPERATOR
```

أي تنفيذ الحالة الاولى '+' case وحيث إنه لا توجد جملة break فقد تم الانتقال إلى الحالة الآتية '-' case ليتم طبع الرسالة :

```
THE '+' IS SUBTRACTION
```

حيث التعبير op لم تتغير قيمته وبذا تم طبع الرسالة غير الصحيحة ونفذ الأمر break بهذه الحالة حيث تم الانتقال إلى نهاية جملة switch ، وبنفس الطريقة تكون النتيجة غير صحيحة في حالة إدخال المؤثرين (* , /) على عكس ذلك فإن النتيجة ستكون صحيحة في حالة إدخال مؤثر (-) .

مثال (3-5-4)

البرنامج الآتي يطبع اسم اللون حسب حالة الحرف الأول من الكلمة الدالة على ذلك اللون .

```
#include <stdio.h>
main()
{
    char color;
    printf("\nGive the color now ==>");
    scanf("%c", &color);
    switch(color)
    {                        /* start switch */
      case 'r' :
      case 'R' : printf("YOUR COLOR IS RED\n");
                 break;
      case 'o' :
      case 'O' : printf("YOUR COLOR IS ORANGE\n");
                 break;
      case 'b' :
      case 'B' : printf("YOUR COLOR IS BLUE\n");
                 break;
      case 'g' :
      case 'G' : printf("YOUR COLOR IS GREEN\n");
                 break;
      default  : printf("YOUR SELECTION [%c] ",color);
                 printf("OUT OF RANGE\n");
    }              /* end switch */
}
```

وفيما يلي النتائج وقت تنفيذ البرنامج بعدد من الاختيارات :-

```
Give the color now ==>g
```

```
YOUR COLOR IS GREEN

Give the color now ==>a
YOUR SELECTION [a] OUT OF RANGE

Give the color now ==>R
YOUR COLOR IS RED
```

بالبرنامج وبعد إدخال الحرف وتخصيصه للمتغير color تتم مقارنته بالحرف 'r' أو الحرف 'R' وإذا ما تحقق الشرط تنفذ الحالة الآتية :-

```
case 'r' :
case 'R' : printf("YOUR COLOR IS RED\n");
        break;
```

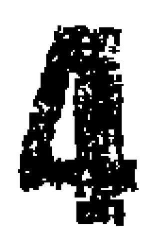
أي تنفيذ جملة الطباعة الداخلة في نطاق هذه الحالة ثم تنفيذ جملة break التي عن طريقها يتم الانتقال إلى قوس الفئة المغلق لجملة switch .

مثال (4-5-4)

اكتب برنامجا كاملا مهمته قراءة المتغير x من النوع الصحيح ثم اعمل الآتي :-

1) إذا كانت x تساوي 6- فاحسب المعادلة y = (x*x)-x .
2) إذا كانت x تساوي 1 أو 5 احسب y = (x*10)/2 .
3) إذا كانت x تساوي 2 فاحسب y = x+x .
4) إذا كانت x تساوي 4 أو 4- فاحسب y = x* x .
5) اطبع الرسالة Sorry data out of range في حالة تغير قيمة x عما ذكر أعلاه .

```
#include <stdio.h>
main()
{
    int x, y;
    printf("\nEnter value of x ==> ");
```



```
    scanf("%d", &x);
    switch (x)
    {
      case -6 : y = (x*x)-x;
                printf("X=%d and  (x*x)-x = %d\n", x, y);
                break;
      case 1 :
      case 5 :  y = (x*10)/2;
                printf("X=%d and (X*10)/2 = %d\n", x, y);
                break;
      case 2 :  y = x+x;
                printf("X=%d and X+X = %d\n", x, y);
                break;
      case -4 :
      case 4 : y = x*x;
                printf("X=%d and X*X = %d\n", x, y);
                break;
      default : printf("Sorry data out of range\n");
    }
}
```

في هذا البرنامج وعوضا عن استخدام جملة if-else تم استخدام جملة switch التي عن طريقها كان البرنامج قصير ومنظم بحيث يمكن متابعته وفهمه بسهولة ويسر ، وقت تنفيذ هذا البرنامج لأكثر من مرة ستكون النتائج مشابهة للآتي:-

```
Enter value of x ==> 5
X=5 and (X*10)/2 = 25

Enter value of x ==> -6
X=-6 and  (X*X)-X = 42

Enter value of x ==> -4
X=-4 and X*X = 16

Enter value of x ==> 3
Sorry data out of range

.

Enter value of x ==> 2
X=2 and X+X = 4
```

مثال (4-5-5)

البرنامج الآتي هو اعادة للبرنامج بالمثال (4-4-1) الذي مهمته إدخال قيمة عددية وطباعة الرسالة المناسبة في حالة ما إذا كان العدد المدخل موجبا أو سالبا أو صفرا باستخدام جملة switch المتداخلة .

```
#include <stdio.h>
main()
{
    int number;
    printf("\nEnter your number ==> ");
    scanf("%d", &number);
    switch ( number > 0 )
    {
      case 1: printf("The %d is positive number", number);
              break;
      case 0: switch ( number < 0 )
              {
                case 1:printf("The %d is negative number", number);
                       break;
                default :printf("The %d is zero number", number);
              }
    }
    printf("\nAll done");
}
```

هنا يتم تقييم الشرط number>0 بجملة switch الخارجية ، فإذا كان الشرط صحيحا يتم تنفيذ جملة الطباعة بالحالة case 1 التي تدل على أن الرقم موجب، وإذا لم يتحقق الشرط تنفذ الحالة case 0 التي يتم بواسطتها تقييم الشرط number<0 بجملة switch الداخلية فإذا ما تحقق الشرط تنفذ الحالة case 1 أي طباعة الرقم سالبا وإلا فتنفذ حالة الإسقاط default أي أن الرقم صفر وفي كل حالة من هذه الحالات يتم طباعة الرسالة :

All done

مثال (4-5-6)

اكتب برنامجا كاملا لاستقبال ثلاثة اعداد من النوع الصحيح تمثل التاريخ بحيث العدد الأول يمثل اليوم والثاني يمثل الشهر أما الثالث فيمثل السنة مع حساب وطباعة التاريخ الآتي للتاريخ المدخل .

التحليل :

- بعد إدخال المعطيات السابقة يجب تحديد الايام بكل شهر كالآتي :-

1) الشهور : 12,10,8,7,5,3,1 بها 31 يوما .
2) الشهور : 11,9,6,4 بها 30 يوما .
3) الشهر 2 يحدد بقسمة السنة على 4 وعليه يمكن معرفة ما إذا كانت السنة كبيسة إذا كان باقي القسمة يساوي صفرا وبالتالي فإن الشهر به 29 يوما أو بسيطة إذا كان باقي القسمة لا يساوي صفرا وبالتالي فإن الشهر به 28 يوما .

```
#include <stdio.h>
#include <process.h>  /*  for exit function  */
main()
{
    int day, month, year, day_in_month;
    printf("\nEnter day ==> ");
    scanf("%d", &day);
    printf("Enter month ==> ");
    scanf("%d", &month);
    printf("Enter year ==> ");
    scanf("%d", &year);
    if( (day<1 || day>31) || (month<1 || month>12) )
     {
       printf("\n***** ERROR DATA *****");
       exit(0);
     }
     printf("THE DAY FOLLOWING ");
     printf("%d/%d/%d",day,month,year);
       /*  find number of days in month  */
     day_in_month = 31;
```

```
        switch(month)
        {
          case 4 : case 6 : case 9 :
         case 11: day_in_month = 30;
                break;
         case 2 : if((year % 4 ==0) && (year != 1900))
                   day_in_month = 29;
                else
                   day_in_month = 28;
        }
        if(day == day_in_month)
        {
          day = 1;
          if(month == 12)
          {
            month = 1;
            ++year;
          }
          else
            ++month;
        }
        else
          ++day;
        printf(" IS %d/%d/%d\n", day, month, year);
}
```

وقت تنفيذ البرنامج وفي حالة إدخال بيانات غير مقبولة يتم الخروج من البرنامج وايقاف تنفيذه عن طريق الدالة exit التي سوف نتناولها بالشرح فيما بعد إن شاء الله ، أما في حالة إدخال بيانات مقبولة ، فسوف ينتج عنه الآتي في حالة تنفيذه أكثر من مرة .

```
Enter day ===>  21
Enter month ===>  1
Enter year ===>  1970
THE DAY FOLLOWING 21/1/1970 IS 22/1/1970

Enter day ===> 5
Enter month ===> 15
Enter year ===> 1999
***** ERROR DATA *****
```

```
Enter day ===> 28
Enter month ===> 2
Enter year ===> 1994
THE DAY FOLLOWING 28/2/1994 IS 1/3/1994

Enter day ===> 31
Enter month ===> 12
Enter year ===> 1985
THE DAY FOLLOWING 31/12/1985 IS 1/1/1986
```



6.4) تمرينات Exercises

1) باستخدام جمل الاشهار والتخصيص الآتية :-

```
int x = 2, y = 4, z = 8, w = 5;
int r, a = 1, b = 0;
```

أوجد قيمة المتغير r لكل من الفقرات الآتية :-

a)
```
if(a)
    z += x+y;
else
    r = z - y;
```

b)
```
if( !a || b && !y )
    z = x+y;
else
 r = z + w*y;
```

c)
```
if(z/y == x)
    z = ++x+y;
else
    r = --z - y;
```

d)
```
if( w!= x+z || b !=a )
    z = --x+y;
else
    r = ++z + y;
```

e)
```
if(2*x <= z && (w*x-z+x))
{
    int a = 3;
    z /= a*x+y;
}
else
    z = a*y+z;
r = z+y;
```

f)
```
if(a == !b && !(x+y == z-x))
    z += w;
else
{
    a += 6;
    w +=a;
}
r = z+w;
```

2) صمم برنامجاً لقراءة قيمة a تمثل درجة الحرارة وحرفاً op يمثل نوع هذه الدرجة (مئوية c أو فهرنهيت f) ثم يحسب f اذا كانت الدرجة مئوية ويحسب c اذا كانت فهرنهيت حيث :

$$f = \frac{9}{5}[32 + a] \qquad c = \frac{5}{9}[a - 32]$$

3) اكتب برنامجا يقرأ ثلاثة امتحانات ويطبع الاكبر فيهم .

4) المطلوب كتابة برنامج يقرأ مرتب البائع ومقدار مبيعاته ثم يحسب النسبة المئوية المكافئته التي هي على النحو الآتي :-.

2%	إذا كانت مبيعاته اقل من أو تساوي ثلاثة اضعاف مرتبه.
3%	إذا كانت مبيعاته أكبر من ثلاثة اضعاف مرتبه .
4%	إذا زادت مبيعاته علي خمسة اضعاف مرتبه .

5) مستخدما جملة switch أوجد قيمة المتغير y حيث :

$$y = \begin{cases} x+10 & \text{if } x > 0 \\ 1000 & \text{if } x = 0 \\ 25-x & \text{if } x < 0 \end{cases}$$

6) اكتب برنامجا لقراءة سطر واحد يحتوي على رقم صحيح A ورقم حقيقي B وحرفي C ثم احسب العمليات A+B , A-B , A*B , A/B اعتمادا على الحرف C الذي قد يكون أحد المؤثرات (+, -, *, /) اطبع الرسالة المناسبة مع الأخذ فى الاعتبار أن الحرف المدخل لم يكن أحد المؤثرات السابقة مستخدما جملة if مرة وجملة switch مرة اخرى .

7) اكتب برنامجا يقوم بإدخال ثلاث قيم صحيحة ، الأولى تمثل اليوم والثانية الشهر والثالثة السنة ، ثم اطبع هذه البيانات مع كتابة اليوم بالحروف .

8) اكتب برنامجا لقراءة ثلاث درجات مع إيجاد مجموع أعلى درجتين .

9) أعد كتابة مثال (4-5-4) مستخدما جملة if .

10) المطلوب كتابة برنامج مهمته استقبال رقم قيد الطالب ودرجاته فى الامتحان الأول والثاني والنهائي ثم يطبع رقم القيد ومجموع درجاته مع حالته استنادا على الآتي :-



الحالة	الدرجة
Fail	اصغر من 50
Pass	أكبر من أو تساوى 50 وأصغر من 65
Good	أكبر من أو تساوى 65 وأصغر من 75
V. Good	أكبر من أو تساوى 75 وأصغر من 85
Exel.	أكبر من أو تساوى 85 وأصغر من أو تساوي 100

11) قم بكتابة برنامج يقرأ رقم صحيح وطباعة نصفه اذا كان الرقم زوجي وطباعة ربعه إذا كان الرقم فردي.

12) صمم برنامجاً لحساب معادلة من الدرجة الثانية :

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

وطباعة الجذرين في حالة التعبير (b^2-4ac) موجباً وطباعة $x = \frac{b}{2a}$ اذا كان التعبير صفراً وطباعة الرسالة المناسبة اذا كان غير ذلك .

13) ما هو الناتج عند تنفيذ الآتي:

a)

```
{    int x = 11, y = 9;
     if( x<10 )
        if( x>y )
           printf("Yes User\n");
        else
           printf("No User\n");
     printf("Good Bye User\n");
}
```

b)

```
{    int x = 11, y = 9;
     if( x==11 )
        if( y<x )
           printf("Yes User\n");
        else
           printf("No User\n");
     printf("Good Bye Use\n");
}
```

5

الفصل الخامس

جمل التكرار

1.5) التكرار *Repetition*

التكرار يعني تنفيذ جملة أو مجموعة من الجمل عددا من المرات وذلك تحت شرط معين يفرضه حل المسألة المعطاة ، وكما نلاحظ في البرامج السابقة التي كتبت أننا لم نتمكن من تكرار عمل هذه البرامج لأكثر من مرة ، عليه في هذا الفصل سوف نتطرق إلى استخدام جمل التكرار مثل for, while التي لها أشكالها المختلفة .

2.5) جملة *While*

لغة C غنية جدا بالأوامر التي تنفذ مجموعة من الجمل عدة مرات ومنها جملة while التي لها الشكل الآتي :-

```
while(condition)
statement;
```

عند تكرار تنفيذ جملة واحدة ، أما في حالة تنفيذ عدة جمل وهذا ما يحدث غالبا فالشكل هو :-

```
while(condition)
{
     statement_1;
     statement_2;
     statement_3;
     . . .
     statement_n;
}
```

اختبر الشرط الذي يجب وضعه بين القوسين () أولا ، فإذا كانت نتيجته true أي له قيمة صحيحة عدا الصفر ، نفذ الجملة المركبة أي المحاطة بين

قوسى الفئة { } ويتكرر هذا الاختبار والتنفيذ حتي يتم تغير الشرط وتصبح قيمته صفرا أي false عندها ينتقل التحكم إلى تنفيذ الجملة الموالية للقوس المغلق } .

مثال (5-2-1)

البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int m;      /*  m as a counter  */
    printf("The output as following\n");
    m = 1;
    while( m <= 5 )
    {
      printf(" M=%d\n", m);
      ++m;
    }
}
```

فيه تم تخصيص قيمة 1 للعداد m قبل الدخول إلى الحلقة التابعة لجملة while ، يأتي بعده الشرط (m<=5) وحيث أنه صحيح عليه يتم تنفيذ الجملتين الواقعتين بين القوسين { } وهما :-

```
printf(" M=%d\n", m);
m += 1;
```

بحيث ينتج عنهما طباعة القيمة 1 للعداد m ثم تزداد قيمته واحدا كل مرة تنفذ فيها هذه الحلقة حتي تصبح قيمته أكبر من العدد 5 عندها تصير قيمة الشرط خاطئة وبالتالي ينتقل التحكم إلى الجملة الموالية للقوس } أي نهاية البرنامج .

وإذا ما نفذ هذا البرنامج سينتج عنه النتائج الآتية :-

```
The output as following
     M=1
     M=2
     M=3
     M=4
     M=5
```

مثال (5-2-2)

يمكن إعادة كتابة البرنامج بطريقة مختصرة كالآتي :-

```
#include <stdio.h>
main()
{
    int m;
    printf("The output as following\n");
    m = 0;
    while( ++m <= 5 )
      printf(" M=%d\n", m);
}
```

ميزة أخرى تضاف إلى لغة C وهى استخدام الزيادة ضمن شرط while حيث تمت زيادة العداد m بالقيمة 1 أولا ثم اختبار الشرط ثانيا وتنفيذ جملة الطباعة الموالية لجملة while وهكذا تتكرر هذه العملية حتي تصبح قيمة m أكبر من 5 عندها تنتهي عملية التكرار ويتوقف البرنامج .

مثال (5-2-3)

ماذا يحدث إذا لم نستعمل القوسين {} بعد جملة while بمثال (5-2-1) ؟ البرنامج الآتي يوضح هذا .

```
#include <stdio.h>
main()
{
    int m;
```

```
    printf("The output as following\n");
    m = 1;
    while( m <= 5 )
        printf(" M=%d\n");
    ++m;
}
```

هنا يكون ناتج تنفيذ هذا البرنامج غير الذي نتوقعه وهو

```
The output as following
M=1
M=1
M=1
. . .
```

والسبب أن جملة while تنفذ جملة الطباعة الموالية لها فقط التي ينتج عنها طباعة M = 1 إلى مالا نهاية لأن المتغير m باقي بالقيمة الإبتدائية 1 ولن يصل إلى جملة زيادة المتغير m بالقيمة 1 بحيث يتجاوز القيمة 5 ليصبح الشرط false وينتهي التكرار .

مثال (5-2-4)

نفذ البرنامج الآتي واستنتج ما الذي يطبعه ؟

```
#include <stdio.h>
main()
{
    int m;
    printf("\nThe output as following\n");
    while( m <= 5 )
    {
      printf(" M=%d\n", m);
      ++m;
    }
}
```

وقت التنفيذ سيطبع البرنامج السطر الآتي فقط :-

```
The output as following
```

والسبب أن المتغير m لم تخصص له أي قيمة عددية قبل الوصول إلى جملة while وبالتالي فإن الشرط لم يكن صحيحا لأن قيمة المتغير m غير معروفة وهذا الخطأ وغيره كثيرا ما يحدث من طرف المبرمج المبتدئ.

قد تستخدم الجملة الفارغة (Empty statement) مع جملة while وذلك بوضع الفاصلة المنقوطة (;) بعدها مباشرة ، البرنامج الآتي يوضح هذا .

```
#include <stdio.h>
main()
{
    short i;
    i = 0;
    while( ++i <= 5 )
      ;
    printf("Value of i now ==>%d", i);
}
```

نلاحظ هنا بينما الشرط (++i<=5) نتيجته صحيحه يتم تنفيذ الجملة الفارغة أي لا شيء ينفذ ويزداد المتغير i بالقيمة 1 وهكذا حتي يصبح الشرط خاطئا عندها ينتقل التحكم إلى جملة الطباعة التي ينتج عنها السطر الآتي :-

Value of i now ==>6

3.5) جملة *while* المتداخلة *Nested while Statement*

قد نحتاج في بعض الأحيان إلى استخدام أكثر من جملة while لتكرار جملة أو أكثر بحيث تكون كل واحدة متداخلة مع الأخرى أي واحدة داخل الثانية .

الشكل العام

```
while(condition_1)      /*    outer while  */
{
      statement11;
      while(condition_2)      /*    inner while    */
      {
         statement_21;
         statement_22;
         ...
         statement_2n;
      }        /*   end inner   */
}     /*   end outer   */
```

مثال (5-3-1)

المطلوب كتابة برنامج لإيجاد مضروب الأرقام من 1 إلى 10 حيث مضروب N يحسب كالآتي :-

$$N!=N(N-1)(N-2)\ldots 1$$

هنا إذا كانت N تساوي 5 مثلا فالناتج هو 120 وعليه يمكن استخدام حلقتين الأولي التي تعمل كعداد من 1 وحتي الرقم 10 وليكن المتغير i ومهمتها تنفيذ الحلقة الثانية التي تحسب مضروب العداد i مع طباعة الناتج وذلك بداية من 1 إلى i عن طريق استخدام العداد k .

```
#include <stdio.h>
main()
{
    short i = 1;
    printf("    I    FACTORIAL\n");
    while ( i <= 10 )      /*  outter while  */
    {
       long factorial = 1;
      short k = 1;
       while( k <= i )    /*  inner while  */
       {
         factorial *= k;
         ++k;
```

```
        }
        printf("\t  %d  \t  %ld\n", i, factorial);
        ++i;
    }
}
```

هنا تم تنفيذ الجملة المركبة التي تقع بين القوسين {} الاولين والتابعين لجملة while الخارجية والتي تم فيها الإعلان عن المتغير factorial من النوع الطويل والمتغير k من النوع القصير طالما أن العداد i لم يتجاوز 10 ، تأتي بعدها جملة while الداخلية ومهمتها إيجاد مضروب قيمة i وذلك من خلال الجملتين

```
factorial *= k;
++k;
```

ويتم تنفيذهما مادام العداد k لم يتجاوز قيمة i .

والناتج هو المشابه للآتي :-

```
I     I FACTORIAL
1        1
2        2
3        6
4        24
5        120
6        720
7        5040
8        40320
9        362880
10       3628800
```

مثال (5-3-2)

يمكن إعادة كتابة البرنامج السابق بصورة أكثر اختصارا ليأخذ الشكل الآتي:-

```
#include <stdio.h>
main()
{
    int i = 0;
    printf("    I    FACTORIAL\n");
```

```
    while ( ++i <= 10 )     /*  outter while  */
    {
      long factorial = 1;
      int k = 0;
      while( ++k <= i )   /*  inner while  */
        factorial *= k;
      printf("\t  %d  \t  %ld\n", i, factorial);
    }
}
```

وفيه لم نستخدم القوسين { } بعد جملة while الداخليـة وبالتـالي فانهـا نفـذت جملة واحدة وهي

```
factorial *= k;
```

مادام العداد k لم يتجاوز العداد i .

مثال (5-3-3)

المطلوب كتابة برنامج للحصول على الشكل الآتي :-

```
*
**
***
****
*****
******
*******
********
*********
**********
```

```
#include <stdio.h>
main()
{
    int i = 0;
    while( ++i <= 10 )
    {
      int j = 0;
      while( ++j <= i )
        printf("*");
      printf("\n");
    }
}
```

مثال (5-3-4)

يمكن التحكم في عملية إنهاء جملة while وذلك بإدخال قيمة عن طريق لوحة المفاتيح ، البرنامج الآتي مهمته استقبال رمز وطباعته بالقيمة العددية المقابلة له بنظام آسكي (ascii) مع وقفه إذا تم إدخال الرمز (?) .

```
#include <stdio.h>
main()
{
    char a;
    printf("\nEnter a character ==> ");
    scanf("%c", &a);
    while( a != '?' )
    {
      printf("[%c] in number is %d", a, a);
      printf("\nEnter a character ==> ");
      scanf("\n%c", &a);
    }
}
```

لاحظ أن دالة الإدخال

```
scanf("\n%c", &a);
```

بدأت برمز القفز إلى سطر جديد (\n) وذلك لإعطاء الفرصة للمنفذ لادخال الرمز الآتي وبهذه الطريقة يمكن الحصول على النتائج الصحيحة في حالة تنفيذ هذا البرنامج وإدخال الرمز المناسب كالآتي :-

```
Enter a character ==>a
[a] in number is 97
Enter a character ==>b
[b] in number is 98
Enter a character ==>A
[A] in number is 65
Enter a character ==>?
```

مثال (5-3-5)

اكتب برنامجا للحصول على متوسط N من الأعداد الحقيقية .

للحصول على هذا المطلوب ينبغي اتباع عمل الآتي :-

1) إشهار متغير من النوع الصحيح ليعمل كعداد وليكن counter مع تخصيص قيمة 0 له .

2) إشهار متغير من النوع الحقيقي ليعمل لحفظ مجموع الأعداد الحقيقية وليكن sum مع تخصيص قيمة 0 له .

3) قراءة عدد الأرقام المطلوب إيجاد متوسطها وليكن المتغير n .

4) تكرار الجمل :

a) قراءة العدد الأول وليكن المتغير number .

b) إضافة هذا العدد إلى المتغير sum .

c) زيادة 1 إلى العداد counter .

مادام قيمة العداد counter لا تتعدي عدد الأرقام n .

5) الحصول على المتوسط average بقسمة المجموع sum على عدد الأرقام n .

6) طباعة المتوسط average .

وفيما يلي البرنامج المعد لتنفيذ هذه الخطوات .

```
#include <stdio.h>
     /* This program calculates the average
          of n numbers using while loop    */
main()
{
     int n, counter = 0;
     float number, average, sum = 0.0;
     printf("\nEnter the size of the list ==> ");
     scanf("%d", &n);   /* get number of values */
     printf("The data are : ");
     while( counter++ < n )
     {
               /* Enter the number and add it to sum  */
               scanf("%f", &number);
               sum += number;
 }
```

```
        average = sum/n;
        printf("The average of all numbers %.3f", average);
    }
```

التنفيذ يعطي الآتي :-

```
Enter the size of the list ==> 7
The data are : 10  -1  3  8  5  -4  9
The average of all numbers = 4.286
```

مثال (5-3-6)

ندرس الآن وجها آخر لاستعمال جملة while فالبرنامج الآتي يقرأ مجموعة من الأعداد الصحيحة ثم يقوم بحساب عدد خانات كل عدد مع تحديد نوع إشارة هذا العدد فمثلا العدد 345- يحتوي على ثلاثة أرقام مع إشارة سالبة

```
#include <stdio.h>
main()
{
    long int num1, num2, k = 0;
    char sign;  short count;
    printf("\nThe input data are: ");
    while( ++k <= 4 )
    {
      count = 0;
      sign = '+';
      scanf("%ld", &num1);
      num2 = num1;
      if( num2 < 0 )
      {
        count++;
        sign = '-';
      }
      while( num2 != 0 )
      {
        num2 /= 10;
        count ++;
      }
      if( num1 < 0 )
        printf("The number %ld has %d ", num1, count-1);
```

```
        else
          printf("The number %ld has %d ", num1, count);
        printf(" digits with [%c] sign\n", sign);
    }
```

التنفيذ وإدخال 4 أعداد كما يلي :-

```
The input data are: 123456789  -55555  -777  654321
```

سيعطي النتائج الآتية :-

```
The number 123456789 has 9 digits with [+] sign
The number -55555 has 5 digits with [-] sign
The number -777 has 3 digits with [-] sign
The number 654321 has 6 digits with [+] sign
```

4.5) جملة do-*while*

هذه الجملة تأخذ الشكل العام :

```
do
{
        statement_1;
        statement_2;
        statement_3;
        . . .
        statement_n;
} while(condition);
```

هذه الجملة مشابة لجملة while وتختلف عنها في أمرين ، أولهما أن جملة do-while تبدأ بتنفيذ جملة واحدة أو عدة جمل ويتم التحقق من الشرط في أسفل الجملة do وثانيهما أنها لابد من تنفيذ الجمل الموجودة بين do و while مرة واحدة على الأقل حتي ولو كان الشرط لم يتحقق ، وعليه يجب أخذ الحيطة عند استخدام هذه الجملة .

مع ملاحظة أنه يجب :-

1) وجود القوسين { } في حالة تنفيذ أكثر من جملة .

2) وجود الفاصلة المنقوطة (;) بنهاية الشرط .

مثال (1-4-5)

خذ مثلا البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int i;
    i = 1;        /* i as a counter */
    do
    {
      printf("%4d", i*i);
      ++i;
    } while( i < 11 ) ;
}
```

وفيه يبدأ البرنامج بتخصيص القيمة 1 للمتغير i الذي يعمل كعداد لجملة do-while ثم طباعة مربع قيمة i مع إضافة العدد 1 إلى المتغير i ويستمر هذا العمل مادام أن i أصغر من 11 .

وبتنفيذ هذا البرنامج سيعطي مربع الأعداد من 1 إلى 10 كالآتي :-

1 4 9 16 25 36 49 64 81 100

أيضاً يتم تنفيذ الجمل الواقعة بين القوسين { } وهما

```
printf("%4d", i*i);
++i;
```

مرة واحدة حتي ولو تغير الشرط وأصبح كالآتي :-

```
while(i>11);
```

مثال (2-4-5)

أعد كتابة البرنامج بالمثال (5-3-5) الذي يحسب متوسط N من الأعداد الحقيقية باستخدام جملة do-while .

```
#include <stdio.h>
  /* Program to compute and printf the average
       of N data values using do-while     */
main()
{
    int n, counter = 0;
    float number, average, sum = 0.0;
    printf("\nEnter the size of the list ==> ");
    scanf("%d", &n);
    printf("The data are : ");
    do
    {
      /* Enter the value and add it to sum  */
      scanf("%f", &number);
      sum += number;
      ++counter;
    } while( counter < n ) ;
    average = sum/n;
    printf("The average of all numbers = %.3f", average);
}
```

إذا ما نفذ هذا البرنامج سيعطي النتائج الآتية :-

```
Enter the size of the list ==> 7
The data are : 10  -1  3  8  5  -4  9
The average of all numbers = 4.286
```

مثال (3-4-5)

تداخل جملة do-while يمكن توضيحه بالبرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int inner, outer = 1;
    do     /*    outer do  */
```

```
      {
        printf("\n");
        printf("\nOuter ==> %d\n", outer);
        inner = 1;
        do      /*  inner do  */
        {
          printf("\nInner --> %d\n", inner);
          ++inner;
        } while( inner < 5 ) ;
        ++outer;
      } while( outer <= 3 ) ;
  }
```

وقت التنفيذ سيكون الناتج كما يأتي:-

```
Outer ==> 1
Inner --> 1
Inner --> 2
Inner --> 3
Inner --> 4

Outer ==> 2
Inner --> 1
Inner --> 2
Inner --> 3
Inner --> 4

Outer ==> 3
Inner --> 1
Inner --> 2
Inner --> 3
Inner --> 4
```

5.5) جملة *for*

جملة for تعتبر إحدى أهم جمل التحكم وهي نستخدمها عندما نكون على علم بتكرار جملة أو مجموعة من الجمل المركبة لعدد محدود من المرات .

الشكل العام

```
for(expression1 ; expression2 ; expression3)
   statement;
next statement;
```

حيث :-

expression1 القيمة الإبتدائية التي تحدد للمتغير على أنه عداد .

expression2 شرط الاستمرار .

expression3 جملة الزيادة أو النقصان في دليل الدورة .

ينفذ التعبير الأول expression1 الذي يمثل القيمة الإبتدائية ومن ثم ينفذ التعبير الثاني expression2 الذي يمثل الشرط فإذا كان هذا الشرط صحيحا (true) عندها تنفذ الجملة statement ثم تتم زيادة التعبير الثالث expression3 الذي يعمل كعداد لهذه الجملة ويعاد بعدها اختبار الشرط مجددا وهكذا يستمر تنفيذ الجملة statement حتي يصير التعبير الثاني أي الشرط خاطئا عندها ينتقل التحكم إلى الجملة الآتية next statement .

أما إذا احتوت جملة for على أكثر من جملة فيكون شكلها

```
for(expression1 ; expression2 ; expression3)
{
      statement_1;
      statement_2;
      statement_3;
      . . .
      statement_n;
}
```

مثال (1-5-5)

يمكن استخدام جملة for لتنفيذ جملة واحدة كما يوضحه البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int i;
    for( i = 1 ; i <= 7 ; i++)
       printf("  I=%d", i);
}
```

في هذا البرنامج استخدمت جملة for التي تتضمن الآتي :-

التعبير الأول ;i = 1 أي تخصيص القيمة الإبتدائية 1 للعداد i .

التعبير الثاني i <= 7 يعني كرر جملة الطباعة إذا كانت قيمة العداد i لم تتجاوز العدد 7 .

التعبير الثالث ++i يعني زيادة المتغير i بالقيمة 1 بعد كل خطوة تنفيذ

وينتج عنها طباعة السطر الآتي وقت تنفيذ البرنامج .

```
I=1  I=2  I=3  I=4  I=5  I=6  I=7
```

مثال (5-5-2)

ميزة أخرى من ميزات لغة C تتمثل في وضع تعبير الزيادة أو النقصان مع تعبير الشرط معا على أن توضع الفاصلة المنقوطة (;) محل التعبير الثالث ، وعليه يمكن إعادة كتابة البرنامج السابق باستخدام جملة for بالشكل الآتي :-

```
for(i= 0 ; ++i <= 7 ; )
```

الذي يعطي نفس النتائج بالبرنامج السابق .

مثال (5-5-3)

يمكن طباعة الناتج السابق تنازليا وذلك بتغيير جملة for كما يلي :-

```
for(i= 7 ; i >= 1 ; i--)
```

وعليه سيكون الناتج مشابهاً للآتي :-

```
I=7  I=6  I=5  I=4  I=3  I=2  I=1
```

حيث خصص العدد 7 للمتغير i وبالتالي يجرى تكرار جملة الطباعة مادام المتغير i ما يزال أكبر من أو يساوي 1 ثم يطرح 1 من العداد i .

مثال (5-5-4)

اكتب برنامجا لإيجاد مجموع الأعداد

sum = 10.0 , 9.5 , ... , 5.5 , 5.0 , 4.5

```
#include <stdio.h>
main()
{
      float a, sum = 0.0;
      for( a = 4.5 ; a <= 10 ; a += 0.5)
        sum += a;
      printf("\nTHE SUM IS %.2f", sum);
}
```

في هذا البرنامج استخدم التعبير الأول الذي فيه إسناد القيمة 4.5 للمتغير الحقيقي a وحيث إن شرط جملة for صحيح فقد جرى تنفيذ جملة sum+=a أي وضع القيمة 4.5 بالمتغير sum ومن ثم زيادة a بالقيمة 0.5 .

مثال (5-5-5)

يمكن إعادة كتابة نفس البرنامج السابق بطريقة أخرى وهي

```
#include <stdio.h>
main()
{
      float a, sum = 0.0;
      for( a = 4.5 ; a <= 10 ; )
      {
         sum += a;
         a += 0.5;
      }
      printf("THE SUM IS %.2f", sum);
}
```

هنا وضعت الفاصلة المنقوطة (;) محل التعبير الثالث a += 0.5 الذي تم وضعه ضمن جمل التنفيذ التابعة لجملة for بين القوسين { } بدلا من أن يكون بداخلها والناتج في الحالتين سيكون مشابها للآتي :-

THE SUM IS 87.00

مثال (6-5-5)

البرنامج الآتي يوضح استخدام التعبيرات الثلاثة في جملـة for كمتغيرات حرفية حيث ينتج عنه طباعة الحـروف الهجائيـة تنازليـا مـن الحـرف Z إلـى الحرف A .

```
#include <stdio.h>
main()
{
    char ch;
    for( ch = 'Z' ; ch >= 'A' ; --ch)
      printf("%c", ch);
}
```

مثال (7-5-5)

لحساب حاصل ضرب جميع الأعداد الصحيحة الزوجيـة مـن 2 إلـى 10 يمكننا استخدام البرنامج الآتي :-

```
#include <stdio.h>
main()
{
     int I, product = 1;
     printf("\n The product of all even numbers ");
     for(i = 2 ; i <= 10 ; i += 2)
     {
        product *= i;
        printf("%d ", i);
     }
     printf("==> %d", product);
}
The product of all even numbers 2  4  6  8  10 ==> 3840
```

بالبرنامج جملة for التي تحتوي على :-

1) التعبير الأول i=2 حتي تكون بداية العداد بالقيمة 2 .

2) مقدار زيادة المتغير i بمقدار 2 للحصول على الأعـداد الزوجيـة وقـد يأخذ الشكل i+=2 أو الشكل i=i+2 .

التي مهمتها تنفيذ الجملتين

```
product *= i;
printf("%d ", i);
```

لذلك يتحتم علينا وضعهما بين القوسين {} حتي يمكن تكرارهما 5 مرات .

البرنامج يولد النتائج الآتية :-

```
The product of all even numbers 2  4  6  8  10 ==> 3840
```

حيث تمت طباعة قيمة المتغير i وهي 2 ,4 ,6 ,8 ,10 والقيمة النهائية لحاصل الضرب وهي 3840 .

مثال (8-5-5)

إذا ما تم الاستغناء عن التعبير الأول فيجب أن يكون قد تم تعريفه قبل تنفيذ جملة for ، عليه يمكن إعادة البرنامج بالمثال السابق على النحو الآتي:-

```
#include <stdio.h>
main()
{
      int i, product = 1;
      printf("\n The product of all even numbers ");
      i = 2;
      for( ; i <= 10 ; i += 2)
      {
         product *= i;
         printf("%d ", i);
      }
      printf("==> %d", product);
}
```

هنا قيمة التعبير الأول ;i=1 وُضِعت قبل جملة for ووضعت الفاصلة المنقوطة (;) بدلا عنها والناتج يكون مشابها تماما لنفس البرنامج بالمثال السابق.

مثال (5-5-9)

انظر إلى البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int i;
    for( i = 2  ; ; i += 2)
      printf("%d  ", i);
}
```

وفيه حذف التعبير الثاني وهو شرط الاستمرار ومن ثم جرى تنفيذ جملة الطباعة إلى مالانهاية وينتج عن ذلك طباعة الآتي :-

2 4 6 8 10

أيضاً يجوز حذف التعبيرات الثلاثة وهي القيمة الإبتدائية والشرط والزيادة أو النقصان ، وعليه تكون جملة for كالآتي :-

```
for( ; ; )
  statement;
```

وفي هذه الحالة تصبح جملة for لانهائية التكرار .

مثال (5-5-10)

المطلوب إعادة البرنامج بالمثال (5-3-5) والذي يحسب متوسط N من الأعداد الحقيقية باستخدام جملة for .

```
#include <stdio.h>
      /* This program calculate the average
              of n numbers using for      */
main()
{
    int n, counter;
    float number, average, sum = 0.0;
    printf("\nEnter the size of the list ==> ");
      /* Get number of values */
    scanf("%d", &n);
```

```
    printf("The data are : ");
    for(counter = 0 ; counter < n ; counter++)
    {
      /*  Enter the number and add it to sum  */
      scanf("%f", &number);
      sum += number;
    }
    average = sum/n;
    printf("\nThe average of all numbers = %.3f", average);
}
```

وهذه هي نتائج تنفيذ هذا البرنامج .

```
Enter the size of the list ==> 7
The data are : 10  -1  3  8  5  -4  9
The average of all numbers = 4.286
```

6.5) جملة for المتداخلة Nested for Statement

جملة for المتداخلة تستخدم كثيرا خصوصا في معالجة المصفوفات وغيرها وقد وصفت بالمتداخلة لأن كل واحدة منها تكون متداخلة في الأخرى كما يأتي:-

```
for(...)
    for(...)
       for(...)
       statement;
```

مثال (1-6-5)

المطلوب كتابة برنامج لطباعة الحروف C, B, A لكل الاحتمالات الممكنة بحيث لايمكن تكرار احد هذه الحروف في المرة الواحدة .

```
#include <stdio.h>
main()
{
    char L, M, N;
    for(L = 'A' ; L <= 'C' ; L++)
```

```
    for(M = 'A' ; M <= 'C' ; M++)
     if( L != M )
       for(N = 'A' ; N <= 'C' ; N++)
        if((L != N) && (M != N))
             printf("\n%c ==> %c ==> %c", L, M, N);
}
```

في البرنامج تم استخدام أكثر من جملة for وفيها التعبيرات الثلاثة من النـوع الحرفـي وينتـج عنهـا الاحتمـالات التـي سـتطبع الحـروف C, B, A كالآتي:-

```
A ==> B ==> C
A ==> C ==> B
B ==> A ==> C
B ==> C ==> A
C ==> A ==> B
C ==> B ==> A
```

مثال(5-6-2)

المطلوب كتابة برنامج يستقبل رقم قيد الطالب ودرجات لأربعة مقررات دراسية مع طباعة رقم القيد ومعدله لفصل به عدد n من الطلبة .

```
#include <stdio.h>
     /* Sample program with nested for */
main()
{
    short i;
    int number_of_student;
    printf("\nEnter number of student ==> ");
    scanf("%d", &number_of_student);
    printf("------------------------");
    for(i = 1 ; i <= number_of_student ; i++)
    {
      long student_number;
      int j, mark, sum = 0;
      float average;
      printf("\nType student #%d ", i);
      printf("and 4 grades: ");
      scanf("%ld", &student_number);
      for(j = 1 ; j <= 4 ; j++)
```

```
      {
        scanf("%d", &mark);
        sum += mark;
      }
      average = sum/number_of_student;
      printf("\nThe student number ");
      printf("%ld has ", student_number);
      printf("the average %.2f\n", average);
    }
}
```

في هذا البرنامج تم استعمال جملة for المتداخلة لعمل المطلوب بحيث :

1) جملة for الأولى أو الخارجية مهمتها حساب معدل درجات الطالب وطباعته مع رقم القيد لعدد n من المرات .

2) جملة for الثانية أو الداخلية مهمتها قراءة رقم الطالب ودرجاته وحساب مجموع هذه الدرجات .

وفيما يأتي المدخلات والمخرجات وقت تنفيذ البرنامج لعدد 4 من الطلبة .

```
Enter number of student ==> 4
------------------------------
Type student #1 and 4 grades:9905001  60  70  50  80
The student number 9905001 has the average 65.00
Type student #2 and 4 grades:9905026  30  45  39  42
The student number 9905026 has the average 39.00
Type student #3 and 4 grades:9805093  55  55  55  55
The student number 9805093 has the average 55.00
Type student #4 and 4 grades:9905999  75  69  81  89
The student number 9905999 has the average 78.50
```

مثال (5-6-3)

اكتب برنامجا مهمته استقبال أطوال أضلاع مثلث ثم :-

1) اطبع "EQUILATERAL" في حالة تساوي الأضلاع .

2) اطبع "ISOSCELES" في حالة متساوي الساقين .

3) اطبع "SCALENE" في حالة اختلاف الأضلاع .

4) اطبع الرسالة ERROR TRIANGLE LENGTH في حالة عدم التوافق مع المذكور أعلاه .

5) كرر الخطوات السابقة مادامت الإجابة بنعم .

```
#include <stdio.h>
main()
{
    int i, n, k;
    float a, b, c;
    char response;
    do
    {
       printf("\nEnter number of triangles you would\n");
       printf("like to see or type ( 0 to stop ): ");
       scanf("%d", &n);
       for(i = 0 ; i < n ; i++)
       {
            printf("\nEnter three sides of triangle: ");
            scanf("%f %f %f", &a, &b, &c);
            switch ( (a<=0) || (b<=0) || (c<=0) )
            {
               case 1 : k = 4; break;
               case 0 : switch ( (a==b) && (b==c) )
                        {
                           case 1 : k = 1; break;
                           case 0 : switch ( (a==b) || (b==c) || (a==c) )
                                    {
                                       case 1 : k = 2; break;
                                       case 0 : k = 3; break;
                                    }
                        }
            }
            switch(k)
       {
            case 1 : printf("A= %.2f,", a);
                     printf("B= %.2f,C= %.2f", b, c);
                     printf(" The triangle is");
                     printf(" EQUILATERAL\n"); break;
            case 2 : printf("A= %.2f,", a);
```

```
                    printf("B= %.2f,C= %.2f", b, c);
                    printf("The triangle is");
                    printf(" ISOSCELES\n"); break;
          case 3 : printf("A= %.2f,B= %.2f", a, b);
                    printf(",C= %.2f", c);
                    printf("The triangle is ");
                    printf(" SCALENE\n"); break;
          case 4 : printf("A= %.2f,B= %.2f", a, b);
                    printf(",C= %.2f", c);
                    printf(" ERROR TRIANGLE LENGTH \n");
                    break;
        }
      }
      printf("\nClassify another triangle (Y/N) ");
      scanf("\n%c", &response);
    } while( (response=='Y') || (response=='y') );
}
```

في هذا البرنامج استخدمنا جملة do-while التي بها تكرار عدد من الجمل وعن طريقها يتم إيقاف تنفيذ البرنامج بإدخال حرف غير الحرف Y أو الحرف y وهي تعمل بمثابة الحلقة الخارجية ، استخدمنا أيضاً جملة for التي تعتبر الحلقة الداخلية ومهمتها تنفيذ جملة إدخال أطوال أضلاع المثلث ومن ثم إيجاد الحالة التي يكون عليها المثلث عن طريق جملة switch .

وإذا ما نفذ هذا البرنامج سيكون هناك نوع من الاتصال بين الحاسب والمستخدم قد يكون كالآتي :-

```
Enter number of triangles you would
like to see or type ( 0 to stop ): 2

Enter three sides of triangle: 3.4   5.1   3.4
A=3.40 , B=5.10 , C=3.40 The triangle is ISOSCELES

Enter three sides of triangle: 3.1   3.1   3.1
A=3.10 , B=3.10 , C=3.10 The triangle is EQUILATERAL

Classify another triangle (Y/N) y

Enter number of triangles you would
```

```
like to see or type ( 0 to stop ): 2

Enter three sides of triangle: 2.5   -1.3   2.5
A=2.50 , B=-1.30 , C=2.50 ERROR TRIANGLE LENGTH

Enter three sides of triangle: 4.1   4.2   4.3
A=4.10 , B=4.20 , C=4.30 The triangle is SCALENE

Classify another triangle (Y/N) N
```

مثال (4-6-5)

المطلوب كتابة برنامج مهمته استقبال عدد صحيح n مع إيجاد كل الأعداد التي يقبل العدد n القسمة عليها وإذا لم يوجد أي عدد اطبع الرسالة المناسبة التي تدل على ذلك .

```
#include <stdio.h>
          /* Program reads numbers and calculates
              their divisors, it terminates when
              the number is less than or equal to zero  */

main()
{
    int n, div, divisor, done;
    printf("\nGive the number please ==> ");
    scanf("%d", &n);
    while( n > 0 )
    {
      printf("\nThe divisors of number");
      printf(" %d are ==> ", n);
      done = 1;
      for(div = 2 ; div < n ; div++)
      {
        if( (n/div)*div == n)
        {
          divisor = div;
          printf("%5d", divisor);
          done = 0;
        }
      }
```

```
    if( done )
    printf("not found it is prime number \n", n);
    printf("\nEnter the number if you would like to see");
    printf("\nanother divisors or ( Type 0 ) to stop : ");
    scanf("%d", &n);
  }
  printf("\nProgram execution terminated good bye");
}
```

بالبرنامج تم إدخال العدد n وعن طريق جملة while طبعت الأعداد التي يقبل العدد n القسمة عليها في كل مرة ويتكرر هذا إلى أن يتم إدخال عدد أقل من أو يساوي صفر عندها تطبع الرسالة المناسبة وينتهي تنفيذ البرنامج ، أما جملة for فهي لإيجاد الأعداد التي يقبل العدد n القسمة عليها .

وهذه هي نتائج البرنامج عند التنفيذ .

```
Give the number please ==> 48

The divisors of number 48 are ==>2   3   4   6   8   12   16   24
Enter the number if you would like to see
another divisors or ( Type 0 ) to stop : 17
The divisors of number 17 are ==> not found it is prime number

Enter the number if you would like to see
another divisors or ( Type 0 ) to stop : 256

The divisors of number 256 are ==>2   4   8   16   32   64   128
Enter the number if you would like to see
another divisors or ( Type 0 ) to stop : 0

Program execution terminated good bye
```

7.5) تمرينات Exercises

1) باستخدام جملة for اطبع الحروف من 'A' إلى 'Z' وذلك على النحو الآتي:-

* حرفا واحدا في كل سطر .
* خمسة حروف بالسطر الواحد .
* كل الحروف في سطر واحد .

2) المطلوب كتابة برنامج مهمته قراءة القيم الآتية :

```
65 37
22 51
19 91
25 25
14 87
74 12
```

ثم يحسب متوسط كل عمودين أولاً ومتوسط كل صف ثانيا مع إيقاف البرنامج إما بعدد السطور أو بادخال قيمة سالبة في نهاية هذه القيم .

3) باستخدام جملة while ، اكتب برنامجا لقراءة n من النوع الصحيح ثم أوجد واطبع الآتي :-

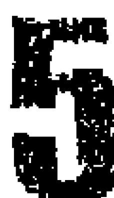
a) $sum1 = 1 + 2 + 3 + ... + n$
b) $sum2 = 1 - 2 + 3 - 4 + ... n$
c) $sum3 = x + x/2! + x/3! + ... + x/n!$

4) أعد كتابة حل التمرين (3) باستخدام جملة for .

5) اكتب برنامجا يحسب حاصل جمع مربعات الأعداد الصحيحة الفردية الواقعة بين عددين صحيحين يتم إدخالهما عن طريق لوحة المفاتيح .

6) المطلوب كتابة برنامج يقرأ متغيرين n, m من النوع الصحيح ثم يحسب قيمة p حيث :

$$P = \frac{n!}{(n-m)!}$$



في حالة (n>m) وذلك باستخدام جمل do-while, while, for .

7) أوجد ناتج الفقرات التالية أولا ثم أعد كتابتها باستخدام جملة while ثانياً.

```
a)  for(y = 2005 ; y < 2011 ; y+=2)
    printf("spring %d", y);
b)  for(j = 10; j >= 4 ; j--)
    printf("J*J=%3d", j*j);
c)  for(a = 1 ; a <= 3 ; a++)
    for(b = a ; b<=5 ; b += 2)
    printf("a*b=%3d", a*b);
d) int i,j,a=1,b=5;
   for(i=1;i<=5;++i)
   {
     for(j=a;j<=b;++j)
          printf("\t%d",j);
     printf("\n");
     a+=5;b+=5;
   }
e) for(i=1;i<=20;++i)
    if(i%5 !=0)
        printf("\t%d",i);
    else
        printf("\t%d\n",i);
```

8) أكتب برنامج مستخداماً الجمل المتداخلة مهمته حساب قيمة W حيث :

$$W = \frac{2x - 3y}{(x-3)(y-6)}$$

حيث x= 5, 4, 3, 2, 1 وقيمة y=2, 4, 6, ..., 10 وذلك لكل قيمة من x .

9) أكتب برنامج يحسب معدل أعمار أسرتك المحصورة بين 7 سنوات و 50 سنه

10) باستعمال الجمل المتداخلة ، اكتب برنامجاً لإخراج الأشكال الآتية :-

```
a)  1  2
    3  4  5
    6  7  8  9
    10 11 12 13 14
```

```
b)  A
    A B
    A B C
    A B C D
```

11) المطلوب كتابة برنامج لقراءة درجات لفصل به N من الطلبة ثم يحسب أعلى واقل درجة في هذا الفصل.

12) المطلوب كتابة برنامج لإيجاد طول أي عدد صحيح موجب يتم ادخاله ، فمثلا العدد 54321 ينتج عنه الطول 5 ، أيضاً طباعة مجموع الأعداد المتكون منها هذا العدد أي 1+2+3+4+5 تساوي 15 .

13) اكتب برنامجا يعكس أي قيمة صحيحة موجبة أو سالبة ، فمثلا القيمة 12345- تصبح 54321- .

14) أوجد ناتج طباعة هذه البرامج باستخدام الورقة والقلم أولاً والحاسب ثانياً.

a)

```
#include <stdio.h>
main()
{
      int m, n;
      for(m = 1; m <= 5; m++)
      {
        printf("m=%d", m);
        for(n = m+2; n < 2*m; n++)
           printf("N=%d", n);
        printf("END\n");
      }
      printf(" BYE");
}
```

b)

```
#include <stdio.h>
main()
{
      int m = 1, n = 3 , s1 = 0 , s2 = 0;
      for( ; m <= n ; )
      {
            int n = 1;
            for(; n< 2 ; )
            {
                  s2 += n*m;
```

```
            n += 2;
            s1 += m+n;
            printf("S1=%d\n", s1);
          }
          ++m;
      }
      printf("S2=%d\n", s2);
  }
```

15) اكتب برنامجا لإدخال رقم الموظف وعدد ساعات العمل الإضافي التي اشتغلها وأجر كل ساعة ثم حساب المبلغ الإجمالي لعدد N من الموظفين في مؤسسة ما ، على أن يكون الناتج مشابها للآتي :-

رقم الموظف	اسم الموظف	الساعات الإضافية	أجرة كل ساعة	المبلغ الإجمالى
–	–	–	–	–
–	–	–	–	–
–	–	–	–	–

16) فصل دراسي به عدد 50 طالبا ، اكتب برنامجا لقراءة رقم الطالب والجنس (1 ذكر ، 2 أنثي) ودرجته في ثلاثة أمتحانات ، المطلوب : إيقاف البرنامج عندما يكون رقم الطالب سالبا ، والمطلوب :-

* حساب متوسط هذه الامتحانات لكل طالب مع طباعة رقمه ومتوسطه والحرف المقابل لمتوسطه على النحو الآتي :-

الحرف	المتوسط
A	من 85-100
B	من 70-84
C	من 55-69
D	من 45-54
F	أقل من 45

* عدد الطالبات وعدد الطلبة والطالبات المتحصلين على التقديرات F , A
* المتوسط العام للفصل مع طباعة رقم قيد الطالب وأكبر متوسط

17) قم بكتابة برنامج يقرأ رقم صحيح ثم يطبع كلمة yes اذا كان هذا الرقم أرمسترونق ويطبع no اذا كان غير ذلك ، الأرقام التالية تعتبر أرمسترونق

$$153 = 1^3+5^3+3^3$$
$$371 = 3^3+7^3+1^3$$

18) المطلوب كتابة برنامج لحساب مجموع الأعداد الزوجية من A الى B
19) صمم برنامجاً كاملا لقراءة قيم للمتغيرات z,y,x وطباعة حاصل جمعهم طالما ان احد هذه المتغيرات غير سالبة.
20) اكتب برنامجاً يطبع عدد القيم الفردية وحساب متوسط القيم الزوجية ، أوقف البرنامج عند إدخال قيمة صفرية .
21) اكتب برنامج يقرأ 10 قيم ويطبع مجموع القيم الفردية والتي تقبل القسمه على 5 وحاصل ضرب القيم الزوجية التي تقبل القسمه على 3 .

22) باستخدام جملة while اكتب برنامجاً كاملا لايجاد طباعة الآتي :-

$$a)\ sum1 = x + \frac{x}{2!} + \frac{3}{3!} + ... + \frac{x}{n!}$$

$$b)\ sum2 = \frac{1}{2!} - \frac{3!}{4} + \frac{5}{6!} - \frac{7!}{8} + ... \pm \frac{n!}{n+1}$$

23) صمم برنامجاً كاملا مهمته حساب قيمة المتغير Y حيث :

$$Y = f(X) + g(X)$$

$$f(X) = X^2 + A$$

$$g(X) = \begin{cases} X^2 + X & \text{if} \quad f(X) > 0 \\ 2X + 5 & \text{if} \quad f(X) \le 0 \end{cases}$$

$$X = -5, -4.5, -4, ..., 4.5, 5$$

24) صمم برنامجاً كاملا لايجاد القاسم المشترك الأكبر لمتغيرين يتم ادخالهما عن طريق المستخدم .

25) المطلوب كتابة برنامج لعيادة طبية بها ثلاثة تخصصات C, B, A ويتردد عليها عدد N من الزوار في اليوم الواحد ، أوجد :-

1) أكبر عدد من الزوار للتخصص B .

2) عدد الزوار المترددين من التخصصين C, A .

3) إيجاد مجموع أيراد العيادة علما بان قيمة الرسوم 10 دينارات للتخصص A ، 15 دينار للتخصصين C, B .

6

الفصل السادس

الجمـــل التفـــريعية

في كثير من الأحيان يتعين على المبرمج تحويل المسار التتابعي لأوامر برنامجه إلى جملة معينة أوالخـــروج من جمـــل الاختيار أو التكرار مثل for, while, switch أو الرجوع إليها أو الخروج نهائيا من البرنامج ، وعليه في هذا الفصل سوف نتطرق إلى الجمل التفرعية التي يجب التقليل من استعمالها .

1.6) جملة go to

هذه الجملة لها الشكل العام :

```
goto lable;
```

تستخدم جملة goto لتحويل سير تنفيذ البرنامج إما بشرط أو بدون شرط وتنفذ حين الوصول إليها ومهمتها توجيه تنفيذ البرنامج والانتقال إلى جملة أخرى عنوانها lable والذي ينطبق عليه نفس شروط المعرف وعادة ما يكون أمام الجملة المراد الانتقال إليها ويكون ذا اسم فريد ويجب التحفظ والتقليل من استعمالها في البرامج لأنها في بعض الأحيان تؤدي إلى جعل البرنامج غير مفهوم وغير واضح هيكليا خصوصا وقت المراجعه والتصحيح .

مثال (6-1-1)

اكتب برنامجا بإستخدام جملة goto لقراءة قيم صحيحة لا يزيد عددها عن 10 ثم أوجد مجموع القيم الموجبة فقط مع إيقاف تنفيذ البرنامج عند إدخال قيمة سالبة أو صفرية .

```
#include <stdio.h>
main()
{
    int i, number, posnumber = 0;
    printf("Please type 10 values :\n");
    for(i = 1 ; i <= 10 ; i++)
    {
      printf("Enter value %d: ", i);
      scanf("%d", &number);
      if( number <= 0 )
      {
        printf("This is negative or zero number\n");
        goto last;
      }
      posnumber += number;
    }
    last :  printf("The sum of positive ");
            printf(" values are: %d", posnumber);
}
```

عند تنفيذ هذا البرنامج ستظهر الرسالة

```
Please type 10 values :
```

الدالة على إدخال 10 قيم وإذا بدأت بإدخال القيم المطلوبة سيكون الناتج كالآتي :-

```
Enter value 1 : 9
Enter value 2 : 4
Enter value 3 : 6
Enter value 4 : 8
Enter value 5 : 1
Enter value 6 : 12
Enter value 7 : -7
This is negative or zero number
The sum of positive values are: 40
```

وكما نلاحظ ، لم يجر إدخال القيم العشر عن طريق جملة الإدخال scanf( ) والتابعة لجملة for والسبب هو شرط جملة if فبمجرد إدخال قيمة غير موجبة يتحقق الشرط وتطبع الرسالة الدالة على أن القيمة المدخلة قيمة سالبة

أو صفرية ثم تحول التنفيذ إلى الجملة المعنونة من قبل جملة goto وهي جملة الطباعة ()printf التي أمام العنوان last المنتهي بالنقطتين (:) حيث نتج عنها طباعة مجموع القيم الموجبة .

2.6) جملة ***break***

إضافة إلى استعمالها مع جملة switch تُسْتَخْدَمُ هذه الجملة للخروج من حلقات التكرار حيث عن طريقها يتم إنهاء التكرار متى وصل التنفيذ إليها .

مثال (6-2-1)

المطلوب إعادة كتابة البرنامج بالمثال (6-1-1) باستخدام جملة break .

```
#include <stdio.h>
main()
{
    int i, number, posnumber = 0;
    printf("Please type 10 values :\n");
    for(i = 1 ; i <= 10 ; i++)
    {
       printf("Enter value %d: ", i);
       scanf("%d", &number);
       if( number <= 0 )
       {
         printf("This is negative or zero number\n");
         break;
       }
       posnumber += number;
      }
      printf("The sum of positive ");
      printf("values are: %d", posnumber);
}
```

تنفيذ هذا البرنامج سيولد نفس النتائج المتحصل عليها من البرنامج المشار إليه وهي :-

```
Please type 10 values :
Enter value 1 : 9
Enter value 2 : 4
Enter value 3 : 6
Enter value 4 : 8
Enter value 5 : 1
Enter value 6 : 12
Enter value 7 : -7
This is negative or zero number
The sum of positive values are: 40
```

فعن طريق جملة break تم إجبار البرنامج على الخروج من الحلقة التكرارية التابعة لجملة for قبل أن تنتهي بموجب شرط جملة if حيث تحقق الشرط وبالتالي ثم تنفيذ جملة الطباعة ، بعده انتقل سير التحكم إلى الجملة الموالية للقوس المغلق } الخاص بجملة for أي طباعة مجموع القيم الموجبة التي تم إدخالها .

3.6) دالة *exit* ( )

هذه الدالة إذا استخدمناها فإن ذلك يعني الخروج من البرنامج كليا كما يدل اسمها بحيث ترجع بالقيمة صفرا إذا كان البرنامج قد نفذ على أحسن ما يرام وبأي قيمة لا تساوي صفرا إذا كان هناك خطأ .

مثال (6-3-1)

ماذا يحدث إذا تمت إعادة البرنامج بالمثال (6-1-1) باستخدام الدالة exit() .

```
#include <stdio.h>
#include <process.h>
main()
{
    int i, number, posnumber = 0;
    printf("Please type 10 values :\n");
    for(i = 1 ; i <= 10 ; i++)
    {
```

```
        printf("Enter value %d: ", i);
        scanf("%d", &number);
        if( number <= 0 )
        {
          printf("This is negative or zero number\n");
          exit(0);
        }
        posnumber += number;
      }
      printf("The sum of positive ");
      printf("values are: %d", posnumber);
}
```

يتم إدخال القيم عن طريق جملة التكرار for وعندما يتحقق شرط جملة if تطبع الرسالة المناسبة ثم تنفذ دالة الخروج exit حيث يتم الخروج ليس من حلقة التكرار for بل الخروج نهائيا من البرنامج وبالتالي لن تنفذ الجملة الموالية للقوس المغلق } لجملة for وعليه لن يطبع البرنامج مجموع الأعداد الموجبة المدخلة ويكون ناتج تنفيذ هذا البرنامج مشابها للآتي :-

```
Please type 10 values :
Enter value 1 : 9
Enter value 2 : 4
Enter value 3 : 6
Enter value 4 : 8
Enter value 5 : 1
Enter value 6 : 12
Enter value 7 : -7
This is negative or zero number
```

4.6) جملة *continue*

جملة الاستمرار تختلف مهمتها عن الجمل التفريعية السابقة فهي تعني الاستمرار في توجيه التحكم إلى القوس المغلق للحلقة أي نهايتها والرجوع

إلى بداية الحلقة وإكمال تنفيذها حتي النهاية .

مثال (6-4-1)

لتوضيح الفرق بين جملة Continue وبقية جمل التفريع ، فإننا نكـرر نفس البرنامج المكتوب بالمثال (6-1-1) وذلك لبيان هذا الفرق .

```
#include <stdio.h>
main()
{
    int i, number, posnumber = 0;
    printf("Please type 10 values :\n");
    for(i = 1 ; i <= 10 ; i++)
    {
      printf("Enter value %d: ", i);
      scanf("%d", &number);
      if( number <= 0 )
      {
        printf("This is negative or zero number\n");
        continue;
      }
      posnumber += number;
    }
    printf("The sum of positive ");
    printf("values are: %d", posnumber);
}
```

تنفيذ البرنامج يعطي النتائج الآتية :-

```
Please type 10 values :
Enter value 1 : 9
Enter value 5 : -3
This is negative or zero number
Enter value 2 : 4
Enter value 5 : 0
This is negative or zero number
Enter value 3 : 6
Enter value 4 : 8
Enter value 5 : -7
This is negative or zero number
Enter value 5 : 1
```

```
Enter value 6 : 12
Enter value 7 : 5
The sum of positive values are: 45
```

هنا إذا ما تم إدخال قيمة سالبة تأتي جملة if ويتحقق شرطها وتطبع الرسالة الدالة على أن القيمة المدخلة قيمة سالبة أو صفرية ثم تنفذ جملة continue ويتحول سير التحكم إلى القوس المغلق } الدال على نهاية الجملة for أي لا يتم إضافة القيمة السالبة أو الصفرية إلى المجموع وأيضاً لن يتوقف إدخال باقي القيم بل يستمر حتي نهاية تنفيذ جملة for ويتم بذلك الحصول على مجموع كل القيم الموجبة فقط .

مثال (6-4-2)

المطلوب كتابة برنامج مهمته استقبال عددين من النوع الحقيقي مع إجراء المؤثرات الحسابية وهي (+ , - , * , /) مستخدما بعض الجمل التي تم شرحها في هذا الفصل .

```
#include <stdio.h>
#include <process.h>      /*  for exit() function  */
#include <conio.h>        /*  for clear screen  */
main()
{
    float num1, num2, result;
    char op;
    clrscr();   /*  clear screen  */
    printf("\n\n");
    printf("\t*------------------------------*\n");
    printf("\t*  A or a ---> Addition       *\n");
    printf("\t*  S or s ---> Subtraction    *\n");
    printf("\t*  M or m ---> Multiplication  *\n");
    printf("\t*  D or d ---> Division   *\n");
    printf("\t*  Any  ---> Exit   *\n");
    printf("\t*------------------------------*\n");
    start : printf("\n\nEnter your option please : ");
        scanf("\n%c", &op);
        if( (op !='A') && (op !='a') &&
```

```
          (op !='S') && (op !='s') &&
          (op !='M') && (op !='m') &&
          (op !='D') && (op !='d') )
        exit(0);
    printf("\nType number one : ");
    scanf("%f", &num1);
    printf("Type number two : ");
    scanf("%f", &num2);
    switch (op)
    {
        case 'A' :
        case 'a' : result = num1+num2;
                   printf("%.2f + %.2f = %.2f",
                          num1, num2, result); break;
        case 'S' :
        case 's' : result = num1-num2;
                   printf("%.2f - %.2f = %.2f",
                          num1, num2, result);
                   break;
        case 'M' :
        case 'm' : result = num1*num2;
                   printf("%.2f * %.2f = %.2f",
                          num1, num2, result);
                   break;
        case 'D' :
        case 'd' : if( num2 == 0 )
                     printf(" ERROR DIVID BY ZERO\n");
                   else
                   {
                     result = num1/num2;
                     printf("%.2f / %.2f = %.2f",
                            num1, num2, result);
                   }
                   break;
    }
    goto start;
}
```

في بداية البرنامج استخدِمت الدالة clrscr() التي مهمتها تنظيف شاشة العرض screen من كل البيانات والمعلومات الموجودة عليها وذلك قبل طباعة قائمة الاختيارات والتي يتبعها الرسالة

Enter your option please :

باختيار الحرف M أو الحرف m الذي يتم إسناده للمتغير الحرفي op ينتـج عنـه تنفيذ ذلـك الاختيـار عـن طريـق جملـة switch أي ضـرب العددين فـي بعضـهما وبالتالي يجرى طبع ناتج هذه العملية وينتقل التحكم إلى العنوان start لتظهر قائمة الاختيار علـى الشاشـة مـرة أخـري وهكـذا يسـتمر تكـرار تنفيـذ البرنـامج بالنسبة لبقية المؤثرات عن طريـق جملـة goto حتـى يتـم إدخـال أي حرف من الحروف غير المذكورة في قائمة الاختيار التي ينتـج عنهـا تحقيـق شرط جملة if وإيقاف البرنامج نهائيا عـن طريـق دالـة الخـروج ()exit وفيمـا يأتي تنفيذ البرنـامج والنتائج المترتبة عليه :-

```
*--------------------------------*
* A or a   ---> Addition        *
* S or s   ---> Subtraction     *
* M or m   ---> Multiplication *
* D or d   ---> Division        *
* Any      ---> Exit            *
*-----------------------------    *
Enter your option please : M

Type number one : 2.5
Type number two : 5
2.50 * 5.00 = 12.50

Enter your option please : a

Type number one : 0.12
Type number two : 0.43
0.12 + 0.43 = 0.55

Enter your option please : d

Type number one : 35
Type number two : 0
ERROR DIVID BY ZERO

Enter your option please : D
```

```
Type number one : 12
Type number two : 2.5
12.00 / 2.50 = 4.80

Enter your option please : X
```

5.6) تمرينات Exercises

1) اذكر الفرق بين جملة break والدالة ()exit ، مع توضيح ذلك بمثال .

2) إذا كان لدينا المقطع الآتي من برنامج غير كامل :

```
if( 5*a+b <= 5*b+b )
      goto done;
    {
      c = b+10;
      goto net_yet;
    }
    done : c = a+3*a*b;
    net_yet : printf("\n%d", c);
}
```

المطلوب تتبعه واستنتاج مخرجاته ، على فرض أن المتغيرات b, a كـانت لها القيم الآتية :-

a) 3, 4   b) 5, 2

c) -2, -7   d) 6, 6

3) افحص البرنامج الآتي وصححه إذا كان به أخطاء وإلا فأوجد ما يطبعه:

```
#include <stdio.h>
main()
{
    int x = 6, y = 2, z = 3;
    no : if( z >= x && x >= y )
             goto yes;
         x += 2;
         z += 3;
         y++;
         printf("\n%d %d %d", x, y, z);
         goto no;
    yes : printf("\ngood bye user");
}
```

4) أعد كتابة التمرين (3) بدون استخدام جملة goto .

5) المطلوب إيجاد ناتج البرامج الآتية أولا واعادة كتابتها ثانيا باستخدام جملة for وبدون استخدام جملة continue .

a)

```
#include <stdio.h>
main()
{
    short i = 3;
    do
    {
      i++;
      printf("\nI=%d", i);
      if( i !=5 )
        continue;
      printf("\n%d", i*i);
    } while ( i != 9 );
    printf("\n\nALL DONE");
}
```

b)

```
#include <stdio.h>
main()
{
    short a, b = 0, s = 0;
    for(a = 5 ; a > 0 ; a--)
    {
      if(a %2 != 1)
        b --;
      else
        b += a;
      s += b;
      continue;
    }
    printf("\na ==>%d", s);
}
```

6) اكتب برنامجا مستخدما فيه جملة goto وجملة if ، ثم استبدلهما مستعملا جملتي switch, while .

7

الفصل السابع

دوال التعامل مع الحرفيات

1.7) المؤشر الحرفي *Character Pointer*

البرامج التي كتبت في الفصول السابقة استخدمنا فيها متغيرات من النوع الحرفي لإدخال رمز واحد فقط ، فالإعلان :

```
char ch;
```

يعني أنه قد تم الإعلان عن المتغير ch من النوع الحرفي ، وعليه يمكن تخزين حرف واحد فقط في موقع هذا المتغير .

وهذا النوع من المتغيرات لا يمكنه تخزين سلسلة حرفية (string) لهذا يمكننا التعمق في إشهار متغيرات أخرى من نوع المؤشر الحرفي الذي يمكن أن يشير إلى بداية السلسلة الحرفية (string) في ذاكرة الحاسب.

الشكل العام

```
type *variable;
```

خذ مثلا

```
char *ch;
```

هنا الرمز * يشير إلى بداية السلسلة المخزنة في العنوان الذي يشير إليه المؤشر ch في حين المتغير ch هو مؤشر من النوع الحرفي .

مثال (7-1-1)

تمعن في البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    char *ch;
    ch = "NICE TO MEET YOU";
    printf("%s\n", ch);
    printf("%c\n", *ch);
}
```

الذي يجب ملاحظته بهذا البرنامج :

1) أنه تم تخصيص عنوان بداية السلسلة وهو الحرف N للمتغير ch .
2) أنه تم إضافة الرمز (\0) إلى نهاية السلسلة لتحديد نهايتها .
3) جملة الطباعة الأولى استُخدِم فيها رمز التوصيف %s لطباعة السلسلة بداية من الحرف الأول N وحتى رمز النهايه (\0) .
4) جملة الطباعة الثانية انتجت طباعة الحرف الأول N فقط الذي يشير إليه المؤشر .

وعليه وقت التنفيذ سيكون الناتج مشابهاً للآتي :-

```
NICE TO MEET YOU
N
```

مثال (7-1-2)

البرنامج الآتي يوضح لنا استخدام المؤشر مع جملة while حيث يقوم بطباعة السلسة حتي نهايتها .

```
#include <stdio.h>
main()
{
    char *a;
    a = "\nWHY DON'T YOU GO NEAR THE WATER ?";
    while( *a != '\0' )
    {
        printf("%c", *a);
```

```
        a++;
      }
  }
```

بالبرنامج وبعد تخصيص سلسلة حرفية للمتغير a بدأت الطباعة من سطر جديد لأن أول رمز بالسلسلة هو رمز القفز إلى سطر جديد بعدها يزداد المؤشر ويطبع الحرف الآتي وهكذا حتي الوصول إلى رمز نهاية السلسلة المنتهية بالرمز (\0) عندها يقف البرنامج ويكون الناتج كالآتي :-

WHY DON'T YOU GO NEAR THE WATER ?

مثال (7-1-3)

ما هو ناتج تنفيذ البرنامج بالمثال السابق إذا ما تغيرت السلسلة وأصبحت بالشكل الآتي :-

WHY DON'T YOU GO \0NEAR THE WATER ?

هنا سيكون الناتج هو :

WHY DON'T YOU GO

أي طباعة السلسلة حرفا بعد حرف حتي نهاية السلسلة المنتهية بالرمز (\0) والموجودة قبل كلمة NEAR .

مثال (7-1-4)

البرنامج الآتي يوضح كيفيت إلحاق سلسلة رمزية بأخرى .

```
#include <stdio.h>
main()
{
    char *name, *name1, *name2;
    int k;
    name1 = "\nThis is the first line";
    name2 = "_While this is the second line";
    name = name1;
    for(k = 1 ; k <= 2 ; k++)
```

```
    {
      while ( *name != '\0' )
      {
        printf("%c", *name);
        name++;
      }
      name = name2;
    }
}
```

في البرنامج جملة for ومهمتها تنفيذ جملة while بقصد طباعة السلسلة الأولى This is the first line حرفاً حرفاً بداية من سطر جديد مع إلحاق السلسلة الثانية While this is the second line_ بنفس السطر أي الناتج سيكون كالآتي :-

This is the first line_While this is the second line

2.7) دوال إدخال الحروف *Functions* For *Input Characters*

فيما سبق من برامج استعملنا دالة scanf() لإدخال البيانات العددية والحروف ، أما الآن فيمكن استخدام دوال أخري خاصة تقوم باستقبال الحرف Character مع التنبيه إلى استخدام الملف stdio.h الخاص بدوال الإدخال والإخراج .

1) دالة *getch* ( )

تستخدم هذه الدالة لإدخال حرف واحد من لوحة المفاتيح وقت تنفيذ البرنامج ويتم ذلك بدون الضغط على مفتاح الإدخال Enter حيث لا يمكن إظهار الحرف المدخل عن طريق هذه الدالة .

مثال (7-2-1)

المطلوب إدخال عدد من الرموز مع طباعتها وإيقاف البرنامج عند إدخال الرمز * .

```
#include <stdio.h>
#include <conio.h>
main()
{
    int  letter;
     clrscr();
    printf("Enter a character or press * to exit ");
    printf("\nPlease enter a letter ==> ");
    letter = getch();
    while( letter != '*' )
    {
      printf("\nThe letter just typed was [%c]", letter);
      printf("\nPlease enter a letter ==> ");
      letter = getch();
    }
}
```

وقت التنفيذ تظهر الرسالة الاولي

Enter a character or press * to exit

تبين كيفية وقف البرنامج يليها ظهور الرسالة الثانية طالبة إدخال الرمز المطلوب وبمجرد الضغط على الرمز المطلوب وليكن ? لا يظهر الرمز المدخل على الشاشة بل يتم طباعته مع الرسالة الموالية مباشرة ودون استخدام مفتاح الإدخال Enter وهكذا حتي يتم إيقاف البرنامج بإدخال الرمز *.

وفيما يأتي ناتج تنفيذ البرنامج :

```
Enter a character or press * to exit
Please enter a letter ==>
The letter just typed was [?]
Please enter a letter ==>
The letter just typed was [A]
Please enter a letter ==>
```

لاحظ أن الرمز * لم يتم عرضه لأنه بمجرد الضغط عليه ينتهي تنفيذ البرنامج .

مثال (7-2-2)

مهمة أخرى لإستخدام الدالة getch() يبينها البرنامج الآتي:

```
#include <stdio.h>
#include <conio.h>
main()
{
      char ch;
      clrscr();
      printf("\nHi there press any key to\n");
      printf("return back to main program");
      getch();
}
```

لاحظنا عند قيامنا بتنفيذ البرامج السابقة ظهور النتائج على شاشة العرض إلا أننا لا نستطيع التمعن فيها إلا إذا ضغطنا على المفتاحين Alt و F5 . ولكن باستخدامنا للدالة getch() التي عادة ما يكون وضعها في آخر جمل البرنامج فإن الحاسب سيعطي فرصة للمستخدم لكي يتمعن في نتائج برنامجه إلى أن يقوم بالضغط على أى مفتاح حيث يتم الرجوع إلى البرنامج الرئيسي .

2) دالة getche ( )

تستخدم هذه الدالة لإدخال حرف واحد وعن طريقها يتم إظهاره على شاشة العرض .

مثال (7-2-3)

أعد كتابة البرنامج بالمثال (7-2-1) باستخدام الدالة getche ( ) .

```
#include <stdio.h>
#include <conio.h>
main()
{
      char letter;
      clrscr();
```

```
    printf("Please press * to stop program");
    printf("\nPlease enter a letter ==> ");
    letter = getche();
    while( letter != '*' )
    {
       printf("\nThe letter just typed was [%c]", letter);
       printf("\nPlease enter a letter ==> ");
       letter = getche();
    }
}
```

عند التنفيذ وإدخال البيانات المعطاة بالمثال (7-2-1-1) سنحصل على النتائج الآتية :-

```
Please press * to stop program
Please enter a letter ==> A
The letter just typed was [A]
Please enter a letter ==> 7
The letter just typed was [7]
Please enter a letter ==> *
```

نلاحظ أن الحرف المدخل A قد تم عرضه عند الإدخال والإخراج معا .

3) دالة () *getchar*

تستخدم هذه الدالة لإدخال حرف واحد أيضاً مثل الدوال السابقة مع إظهار الحرف المدخل على شاشة العرض .

مثال (7-2-4)

عن طريق البرنامج الآتي يتم إدخال وطباعة حرف واحد .

```
#include "stdio.h"
main()
{
    char letter;
    printf("Please enter a letter ==> ");
    letter = getchar();
    printf("\nThe letter just typed was [%c]", letter);
}
```

تظهر الرسالة ويتم إدخال الحرف B مثلا

Please enter a letter ==> B

وبالتالي عرضه كالآتي :-

The letter just typed was [B]

مثال (5-2-7)

راقب تنفيذ البرنامج الآتي عند تكرار الدالة أكثر من مرة .

```
#include <stdio.h>
main()
{
    int i;
    char letter;
    for(i = 0 ; i <= 3 ; i++)
    {
      printf("\nPlease enter a letter ==> ");
      letter = getchar();
      printf("The letter just typed was %c", letter);
    }
}
```

وقت التنفيذ يكون الناتج غير ما نتوقع وهو كالآتي :-

Please enter a letter ==> R
The letter just typed was R
Please enter a letter ==> The letter just typed was

Please enter a letter ==> F
The letter just typed was F
Please enter a letter ==> The letter just typed was

حيث تستقبل الدالة الحرف الأول المدخل R ويُطْبَعُ وبعدها لا تقوم الدالة باستقبال الحرف المدخل الآتي والسبب أنها تأخذ رمز الانتقال إلى سطر جديد (\n) ومن ثم يطبع هذا الرمز على هيئة فراغ إذا ما استخدم رمز التوصيف الحرفي (%c) عند الطباعة ، وإذا ما استبدل التوصيف (%c) بالتوصيف (%d)

سيطبع الرمز(\n) بالقيمة المقابلة وهي 10 ، أي نلاحظ أن الدالة getchar() تم تكرارها أربع مرات ولم تستقبل إلا حرفين فقط نتيجة للسبب المذكور أعلاه.

3.7) دوال إخراج الحروف Functions For Output Characters

هناك دالتان مهمتهما إخراج الحرف وهما putch() ، putchar() وهما تختلفان عن دالة الإخراج printf() لأنهما تستعملان بدون استخدام التوصيف في عملية الإخراج .

مثال (7-3-1)

البرنامج الآتي يبين استخدام هاتين الدالتين .

```
#include <stdio.h>
#include <conio.h>
main()
{
     char ch1, ch2;
     ch1 = 'B';
     ch2 = 'X';
     clrscr();
     putchar('\n');
     putch(ch1);
     putchar('\n');
     putchar(ch2);
}
```

بعد الانتقال إلى سطر جديد عن طريق putchar يتم إخراج الحرف B يلي ذلك الانتقال إلى السطر الآتي وإخراج الحرف X ، أي تكون المخرجات كما يأتي:-

```
B
X
```

4.7) دوال إدخال وإخراج الحرفيات *I*nput *O*utput *S*tring

1) دالة *gets* ( )

دالة scanf() التي تم استخدامها مع البرامج سابقا تقوم بقراءة البيانات العددية سواء الحقيقية أو الصحيحة أو حرف واحد.

مثال (7-4-1)

ماذا يحدث إذا ما استُخدِمت دالة scanf() لإدخال سلسلة حرفية ؟ .

سنحصل على الإجابة بعد تنفيذ البرنامج الآتي :-

```
#include <stdio.h>
main()
{
  char  str1[50];
  printf("\nEnter string please ==>");
  scanf("%s", str1);
  printf("\nThe string read with cin was ==> %s", str1);
}
```

عند ظهور الرسالة

Enter string please ==>

على الشاشة ، وعلى فرض أنه تم إدخال السلسلة الآتية :-

WHY AREN'T YOU AT WORK TODAY ?

بالرغم من استخدام التوصيف (%s) مع الدالتين scanf() , printf() تمت طباعة كلمة WHY فقط وهذا بطبيعة الحال غير صحيح والسبب أنه عند استخدام الدالة scanf() يتم إسناد الحروف بداية من أول السلسلة وحتي الوصول إلى أول فراغ للمتغير str1 فقط أي يكون الناتج هو :

The string read with scanf was ==> WHY

مثال (7-4-2)

البرنامج الآتي يوضح التعامل مع الدالة gets() التي تعني (get string) عند إدخال السلسلة الحرفية حتي وإن احتوت على فراغات .

```
#include <stdio.h>
main()
{
  char  str1[50];
  printf("\nEnter string ==>");
  gets(str1);
  printf("\nThe string was ==> %s", str1);
}
```

وقت التنفيذ إذا كانت المدخلات كالآتي :-

Enter string ==> WHY AREN'T YOU AT WORK TODAY ?

تكون المخرجات كالآتي :-

The string was ==> WHY AREN'T YOU AT WORK TODAY ?

أي أنه تمت طباعة كل السلسلة بما فيها الفراغات عن طريق الدالة gets() وبدون استخدام التوصيف (%s) معها .

2) الدالة *puts*

هذه الدالة هي المقابلة للدالة gets() وتستعمل للتعامل مع إخراج البيانات من نوع السلسلة الحرفية عندما لاتكون هناك حاجة لاظهار الحرفيات بتشكيل معين .

مثال (7-4-3)

البرنامج الآتي هو إعادة للبرنامج في المثال (7-4-2) باستخدام دالتي الإدخال والإخراج معاً .

```
#include <stdio.h>
main()
{
  char  str1[50];
  puts("Enter string please ==>");
  gets(str1);
  puts("The string read with gets was ==> ");
  puts(str1);
}
```

وقت التنفيذ وبعد ظهور الرسالة :

```
Enter string please ==>
```

وإدخال السلسلة :

```
WHY AREN'T YOU AT WORK TODAY ?
```

سيكون الرد هو :-

```
The string read with gets was ==>
WHY AREN'T YOU AT WORK TODAY ?
```

حيث تمت طباعة السلسلة بداية من سطر جديد ، يمكن أيضاً استخدام الدالة puts() لإخراج الحرفيات مثل :

```
puts("THIS IS TRIPOLI LIBYA");
```

5.7) دوال معالجة الحرفيات *String Manipulation Functions*

قبل الدخول في شرح هذه الدوال ، يجدر بنا التنويه بإستخدام الملف <string.h> الذي يحتوى على تعريفات لهذه الدوال .

1) دالة القياس *strlen* Function

هذه الدالة هي اختصار للعبارة (String Length) ، وتستخدم لإيجاد طول السلسلة الحرفية التي تقع قبل الرمز (\0) حيث ينتج عنها قيمة عددية صحيحة، ولها الشكل العام الآتي :-

```
strlen(string1);
```

حيث string1 متغير من نوع السلسلة string .

مثال (7-5-1)

الأمر

```
strlen("HOW ARE YOU ");
```

به سلسلة تحتوى على 11 رمزا أي 9 حروف ومسافتين خاليتين .

2) دالة الوصل *strcat* Function

وهي اختصار للعبارة (String Concatenation) وتستخدم لوصل سلسلة حرفية بأخرى وتأخذ الشكل العام الآتي :-

```
strcat(string1,string2);
```

حيث string1 , string2 متغيران من النوع الحرفي ، وهي تعني وصل السلسلة الحرفية string2 عند نهاية السلسلة string1 وعليه وقت التعامل مع هذه الدالة يجب حجز الطول المناسب للمتغير string1 لأنه سوف يحتوي على طول السلسلتين معا بعد عملية الوصل .

مثال (7-5-2)

البرنامج الآتي يبين كيفية استخدام الدالتين strlen , strcat .

```
#include <conio.h>
#include <string.h> /* definitions of strcat, strlen functions */
#include <stdio.h>

   /* Program using strcat and strlen functions */
main()
{
    char *string1 , *string2;
    clrscr();
    printf("Enter string one :");
    gets(string1);
    printf("Enter string two :");
    gets(string2);
```

```
    strcat(string1, string2);
    printf("\n\nString one and two together ==> %s", string1);
    printf("\nIt has %d characters.", strlen(string1) );
    getch();
    return 0;
}
```

عند تنفيذ هذا البرنامج وإدخال السلسلة الأولى وهي We are in وتخزينهـا بالمتغير الأول string1 والسلسلة الثانية TRIPOLI-LIBYA بـالمتغير string2 كما يأتي :-

```
Enter string one :We are in
Enter string two : TRIPOLI-LIBYA
```

عندها يتم تنفيذ الدالة strcat وبالتالي ضم محتوي المتغير الثاني فـي نهايـة محتوي المتغير الأول مع التخزين ، وأخيراً طباعة هذا المتغير أولا ثم تحديد وطباعة طوله كالآتي :-

```
String one and two together ==>We are in TRIPOLI-LIBYA
It has 23 characters.
```

3) دالة الوصل حتى n حرف *strncat* **Function**

توجد أيضاً دالة أخرى باسم strncat وهي مشابة للدالة strcat من حيث الوظيفة التي شكلها هو :

```
strncat(string1, string2, n);
```

وتقوم بإضافة n حرفا بداية من أقصي يسار السلسلة الحرفيـة string2 إلـى نهاية السلسلة الحرفية string1 .

مثال (7-5-3)

إذا ما أعيدت كتابة البرنامج بالمثال السابق مع تغيير الدالـة strcat بالدالـة strncat كما يأتي :-

```
strncat(string1,string2, 7);
```

ونفّذ البرنامج وأُدْخِلَت نفس المعطيات السابقة :

```
Enter string one :We are in
Enter string two : TRIPLOI-LIBYA
```

سيكون الناتج مشابها للآتي :-

```
String using strncat ==> We are in TRIPOLI
It has 17 characters.
```

حيث أُخِذَت الحروف السبعة الأولى من السلسلة TRIPOLI-LIBYA الموجودة بالمتغير string2 وهي TRIPOLI وإلحاقها في آخر السلسلة الحرفية الموجودة بالمتغير string1 .

4) دالة النسخ *strcpy* **Function**

هذه الدالة هي اختصار للعبارة string Copy وشكلها العام :

```
strcpy(string1,string2);
```

وتستخدم لنسخ السلسلة الحرفية المخزنة في المتغير string2 إلى المتغير string1 بما فيها رمز النهاية (\0) وعليه يفقد المتغير string1 القيمة التي قبل النسخ .

مثال (7-5-4)

العبارة :

```
strcpy(str,"THIS IS A TEST");
```

تعني نسخ السلسلة THIS IS A TEST وتخزينها في المتغير str الذي يجب أن يكون مناسبا من حيث النوع والطول .

مثال (5-5-7)

فيما يأتي برنامج يوضح استخدام دالة النسخ strcpy .

```
#include <conio.h>
#include <string.h>
#include <stdio.h>
/* Program using strcpy() function */
main()
{
    char *string1, *string2;
        clrscr();
    printf("Enter string one :");
    gets(string1);
    printf("Enter string two :");
    gets(string2);
    strcpy(string1, string2);
    printf("\nString one after strcpy");
    printf(" ==> %s", string1);
}
```

إذا ما نفذ هذا البرنامج وأُدخلت البيانات المناسبة سيعطي الناتج الآتي :-

```
Enter string one : This is
Enter string two :a COMPUTER
String one after strcpy ==> A COMPUTER
```

5) دالة نسخ n حرف *strncpy* **Function**

أيضاً هناك دالة أخري لنسخ السلسلة الحرفية وهي strncpy التي تأخذ الشكل الآتي :-

```
strncpy(string1,string2,n);
```

أي نسخ n من الحروف بالضبط من المتغير string2 إلى المتغير string2 وينتهي النسخ عند الحصول على النهاية (\0) في حالة ما إذا كان المتغير string2 يتضمن أقل من n حرفا ، أما في حالة أن المتغير string2 يتضمن n

حرفا أو أكثر ، عندها ينسخ n حرفا فقط وقد لا تنتهي السلسلة الحرفية المخزنة في المتغير string1 بالرمز (\0) .

مثال (7-5-6)

البرنامج الآتي يبين استخدام هذه الدالة .

```
#include <conio.h>
#include <string.h>
#include <stdio.h>
/*  Program using strncpy() function  */
main()
{
    char *string1, *string2;
    clrscr();
    printf("Enter string one :");
    gets(string1);
    printf("Enter string tow :");
    gets(string2);
    strncpy(string1, string2, 4);
    printf("\nString one after strncpy");
    printf(" ==> %s", string1);
}
```

تنفيذ هذا البرنامج وإدخال البيانات سينتج عنه النتائج الآتية :-

```
Enter string one :stay please
Enter string two :LOOK
String one after strncpy ==> LOOK please
```

في هذه الحالة كانت قيمة المتغير n تساوي 4 وهي مساوية لطول السلسلة التي يتضمنها المتغير string2 ، عليه تم نسخ الحروف الأربعة كلها من المتغير string2 ووضعها في نفس عدد الخانات الأربع الأولى في المتغير string1 .

أما إذا ما غيرنا قيمة 4 لتصبح 5 بالدالة strncpy ، فالناتج يكون كما يأتي:-

```
Enter string one :stay please
Enter string two :LOOK
String one after strncpy ==> LOOK
```

في حالة ما إذا كان عدد الحروف المطلوب نسخها يساوي 5 أحرف أو أكثر ، والمتغير string2 يتضمن أقل من n حرفا ، عندها يتم نسخ الحروف المعنية إلى string1 من البداية مع إضافة الرمز (\0) حتى يصبح عدد الحروف المنقولة يساوي n .

6) دالة المقارنة *strcmp* Function

وهي اختصار للعبارة string compare ومهمتها مقارنة متغيرين من النوع الحرفي حيث ينتج عنها قيمة صحيحة ، والشكل العام لها :

```
strcmp(string1,string2);
```

فالدالة strcmp تقارن بين قيمة السلسلتين string1, string2 من حيث ترتيبهما في نظام الشفرة آسكي (ascii) ، انظر ملحق (1) والرجوع بعدد :

1) أكبر من الصفر إذا كان ترتيب string1 أكبر من ترتيب string2 .
2) يساوي صفرا إذا كانت قيمة string1 تكافئ قيمة string2 .
3) أقل من الصفر إذا كان ترتيب string1 أقل من string2

وتتم عملية مقارنة السلسلة التي تبدأ بالحرف 'A' والسلسلة الأخرى التي تبدأ بالحرف 'B' وينتج عنها عدد أقل من الصفر ، والسبب أن 'A' تأتي قبل 'B' من حيث الترتيب في نظام آسكي انظر ملحق (1)، وهذا يحدث أيضاً عند مقارنة الكلمة "fat" مع الكلمة "cat" لأن حرف 'c' أصغر من حرف 'f' ، مع الأخذ في الاعتبار أن الحروف الكبيرة تكون أصغر من الحروف الصغيرة .

مثال (7-5-7)

البرنامج الآتي يبين استخدام دالة المقارنة strcmp لسلسلتين .

```
#include <conio.h>
#include <string.h>
#include <stdio.h>
/* Program using strcmp function */
main()
{
    char *string1, *string2;
    int result;
    clrscr();
    printf("Enter string one :");
    gets(string1);
    printf("Enter string two :");
    gets(string2);
    result = strcmp(string1, string2);
    printf("\nString one after strcmp");
    printf(" ==> %d", result);
}
```

عند تنفيذ البرنامج السابق وإدخال البيانات سيكون الناتج كالآتي :-

```
Enter string one :ABCDIC CODE
Enter string two :abcdic code
String one after strcmp ==> -32
```

نلاحظ أن الدالة رجعت بقيمة سالبة ، لأن السلسة الأولى عند المقارنة هي أقل من السلسلة الثانية حيث الحروف الكبيرة أصغر من الحروف الصغيرة .

فإذا ما نفذ البرنامج مرة أخرى وتم إدخال البيانات كما يلي :-

```
Enter string one :abcdic code
Enter string two :abcdic code
String one after strcmp ==> 0
```

فالقيمة المرجعة هي صفر ، وهي تدل على أن السلسلتين متطابقتان .

أما في حالة إدخال سلسلتين إحداهما أكبر من الثانية في الترتيب ، ستكون قيمة الدالة المرجعة أكبر من الصفر كما يأتي :-

```
Enter string one :abcdic code
Enter string two :ABCDIC CODE
String one after strcmp ==> 32
```

7) دالة مقارنة n حرف *strcncmp* **Function**

هذه الدالة مشابة لدالة المقارنة strcmp التي لها الشكل :

```
strncmp(string1,string2,n);
```

فهي تقارن حتي n حرف من السلسلة بالمتغير string2 .

مثال (7-5-8)

توضيح عمل هذه الدالة يبينه البرنامج الآتي .

```
#include <conio.h>
#include <string.h>
#include <stdio.h>
/* Program using strncmp function */
main()
{
    char *string1, *string2;
    int result;
    clrscr();
    printf("\nEnter string one :");
    gets(string1);
    printf("Enter string two :");
    gets(string2);
    result = strncmp(string1, string2, 4);
    printf("\nString one after strncmp");
    printf(" ==> %d", result);
}
```

عن تنفيذ البرنامج وإدخال البيانات سيعطي النتائج الآتية :-

```
Enter string one :HARD WORK
Enter string two :HARD WARE
String one after strncmp ==> 0
```

فقد تمت مقارنة الحروف الأربعة الأولى من السلسلتين ، والنتيجة تطابق تام ، وعليه أرجعت الدالة strncmp قيمة صفر ، وعند تنفيذ نفس البرنامج السابق مع إدخال السلسلتين على النحو الآتي :-

```
Enter string one :HARD WORK
Enter string two :hard WARE
```

سيكون الرد كالآتي :-

```
String one after strncmp ==> -32
```

عند المقارنة وجد أن الحروف الأربعة الأولى من السلسلة الأولى أقل من الحروف الأربعة الأولى من السلسلة الثانية ، وعليه كانت النتيجة سالبة .

أخيرا ماذا يحدث عند إدخال السلسلتين الآتيتين :-

```
Enter string one :haRD WORK
Enter string two :hARD WARE
String one after strncmp ==> 32
```

هنا تمت مقارنة الحروف haRD مع الحروف hARD حرفا حرفا وبما أن الحرف 'h' هو نفس الحرف في السلسلتين ولكن الفرق بين الحرفين في الخانة الثانية حيث الحرف 'a' أكبر من الحرف 'A' وعليه كانت النتيجة موجبة .

6.7) دوال تبديل الحرفيات *String Alteration* **Functions**

توجد بعض الدوال مهمتها تبديل الحرفيات (Sstrings) إلى أعداد ، الجدول الآتي يبين البعض منها مع الأخذ في الاعتبار استعمال الملف <math.h> معها.

الدالة	معناها
atof()	تبديل الحرف في الشفرة ascii إلى عدد حقيقي مضاعف double
atoi()	تبديل الحرف في الشفرة ascii إلى عدد صحيح int
atol()	تبديل الحرف في الشفرة ascii إلى عدد طويل long
itoa()	تبديل العدد الصحيح int إلى النوع الحرفي
ltoa()	تبديل العدد الصحيح long إلى النوع الحرفي

مثال (7-6-1)

البرنامج الآتي يوضح كيفية استخدام إحدى هذه الدوال وهي دالة atof .

```
#include <conio.h>
#include <math.h>  /*  definitions of atof function  */
#include <stdio.h>
/*  Program using atof function  */
main()
{
    double result;
    char string1[80], string2[80];
    clrscr();
    printf("Enter string one :");
    gets(string1);
    printf("Enter string two :");
    gets(string2);
    result = atof(string1)*atof(string2);
    printf("\nThe product of %s * %s ", string1, string2);
    printf("\nusing atof function = %.3lE", result);
}
```

عند التنفيذ وإدخال القيم المناسبة :

```
Enter string one :3.5
Enter string two :7.5
```

سيظهر الآتي :-

The product of 3.5 * 7.5
using atof function = 2.625E+01

أي تم تبديل الحروف تبعا لنظام آسكي (ascii) إلى عدد حقيقـي من النـوع المضاعف double .

كما يمكن إدخال البيانات على هيئة أرقام يتبعها حروف كما يأتي:-

Enter string one :3.5 Times
Enter string two :7.5 Equal

هنا لن تؤثر الحروف على النتيجة وسيكون الناتج مشابها للسابق .

مثال (7-6-2)

في البرنامج الآتي يتم إدخال رقم صحيح مع تحويله إلى الرقم المقابل فـي النظامين الثماني والستة عشر عن طريق الدالة itoa() .

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
/* Program using itoa function */
main()
{
    char ch[10];
    int num;
    clrscr();
    printf("Type integer number : ");
    scanf("%d", &num);
    itoa(num, ch, 8);
    printf("\nThe number %d \n\t = ", num);
    printf(ch);
    printf(" in octal system\n\t = ");
    itoa(num, ch, 16);
    printf(ch);
    printf(" in hexadecimal system");
    getch();
}
```

بهذا البرنامج تم استخدام الدالة itoa التي ينتج عنها مؤشر حرفي بالشكل الآتي :-

```
itoa(num, ch, 8);
```

حيث :

العنصر الأول num يمثل العدد المراد تبديله .

العنصر الثاني ch يمثل الحرف الناتج

العنصر الثالث 8 يمثل العدد المطلوب وهو النظام الثماني .

عموما فإنه عند التنفيذ تظهر رسالة على شاشة نظيفة تتيح لك إدخال رقم صحيح :

```
Type integer number : 165
```

وفيها تم إدخال الرقم 165 في النظام العشري وبالتالي جاءت الدالة itoa الأولى التي قامت بتحويل هذا الرقم إلى ما يقابله في النظام الثماني وهو 245 والثانية التى حولت نفس الرقم المدخل إلى ما يقابله بالنظام الستة عشرى وهو a5 وفيما يلي نتائج تنفيذ هذا البرنامج :-

```
Type integer number : 165
The number 165
                    = 245 in octal system
                    = a5  in hexadecimal system
```

7.7) دوال معالجة الحرف *Character* **Manipulation Functions**

توجد في لغة C بعض الدوال الأخرى مهمتها معالجة متغيرات من نوع الحرف character التي تحتاج إلى الملف <ctype.h> لأنها مَعَرَّفَة في هذا الملف .

1) دوال اختبار الحرف *Character Testing Functions*

وتقوم هذه الدوال باختبار الحرف ، حيث ترجع بقيمة غير صفرية إذا ما كان الحرف رقما أو حرفا أو فراغا أو علامة ترقيم وهكذا ، أما إذا كان غير ذلك فترجع بالقيمة صفراً ، الجدول الآتي يعرض البعض منها .

الدالة	معناها
isalnum (x)	هل x حرف أو رقم ؟
isalpha (x)	هل x حرف أبجدي ؟
isascii (x)	هل x تناظر أحد رموز الشفرة ASCII ؟
iscntrl (x)	هل x رمز اختبار أو تحكم ؟
isdigit (x)	هل x رقم ؟
isgraph (x)	هل x أحد الحروف المطبوعة ما عدا الفراغ ؟
islower (x)	هل x حرف صغير؟
isprint (x)	هل x قابلة للطباعة ؟
ispunct (x)	هل x علامة ترقيم مثل الشارحة والفاصلة ؟
isspace (x)	هل x فراغ ؟
isupper (x)	هل x حرف كبير ؟
isxdigit (x)	هل x في النظام السادس عشر(0-F) ؟

مثال (7-7-1)

البرنامج الآتي يستقبل متغيرا من النوع الحرفي ثم يقوم بتنفيذ الدوال isxdigit, isalpha, isalnum حيث كل واحدة لها مهمة حسب الجدول المذكور سالفاً.

```
#include <conio.h>
#include <process.h>
#include <stdio.h>
#include <ctype.h>  /*  for isalnum, isalpha and isxdigit functions  */
   /*  Program using isalnum, isalpha and isxdigit functions  */
main()
{
    char ch;
    int answer, op;
    for( ; ; )
    {
      clrscr();
      printf("\nType 1 to get isalnum function");
      printf("\n        2 to get isalpha function");
      printf("\n        3 to get isxdigit function");
      printf("\n        4 to exit ");
      printf("\n\nYour selection please : ");
      scanf("%d", &op);
      switch (op)
      {
        case 1 : printf("\nEnter character : ");
                 ch = getche();
                 answer = isalnum(ch);
                 if( answer )
                      printf("\n%c is alphanumeric", ch);
                 else
                      printf("\n%c is not alphanumeric", ch);
                 break;
        case 2 : printf("\nEnter character : ");
                 ch = getche();
                 answer = isalpha(ch);
                 if( answer )
                      printf("\n%c is a letter", ch);
                 else
                      printf("\n%c is not a letter", ch);
                 break;
        case 3 : printf("\nEnter character : ");
                 ch = getche();
                 answer = isxdigit(ch);
                 if( answer )
                      printf("\n%c is a hexadecimal", ch);
                 else
                      printf("\n%c is not a hexadecimal", ch);
```

```
                break;
            case 4:  exit(0);
        }
        getch();
    }
}
```

إذا ما نفذ هذا البرنامج ستظهر قائمة بالاختيارات المتاحة فيه مع إدخال الرقم المناسب وليكن 1 كما يلي :-

```
Type 1 to get isalnum function
     2 to get isalpha function
     3 to get isxdigit function
     4 to exit
Your selection please : 1
```

هنا ينتقل التحكم إلى تنفيذ الدالة isalnum عن طريق جملة switch التي عن طريقها يتم المطالبة بإدخال حرف معين وليكن Y يتبعها كالاتي :-

```
Enter character : Y
Y is alphanumeric
```

وبنفس الطريقة يمكن تنفيذ هذه الدالة في حالة إدخال الرقم 5 أو الرمز * مع الإجابة في كل حالة كما يلي :-

```
Enter character : 5
5 is alphanumeric

Enter character : *
* is not alphanumeric
```

وإذا تم إدخال الاختيار 2

```
Your selection please : 2
```

فسيتم استقبال متغير من النوع الحرفي ثم التعرف عما إذا كان هذا المتغير حرفا أبجديا وذلك بإستخدام الدالة isalpha() ومع إدخال الحرف Y والرقم 5 سيكون الرد في كل حالة كالآتي :-

Enter character : Y
Y is a letter

Enter character : 5
5 is not a letter

كما نلاحظ أن المتغير ch يعتبر حرفا أبجدياً في حالة تَحَقُقٌ الشرط الآتي:-

((ch>='A' && ch<='Z') || (ch>='a' && ch<='z'))

أخيراً في حالة تنفيذ دالة isxdigit أي لمعرفة هل المتغير المدخل هو في نظام الستة عشر أو لا ، عليه يكون الرد بإدخال الاختيار 3 كما يلي :-

Your selection please : 3

يتبعها إدخال الحرف B

Enter character : B

سيرد الحاسب بالآتي :-

B is a hexadecimal

أما إذا ما أدخل الحرف T مثلا

Enter character : T

فعندها تكون الإجابة :

T is not a hexadecimal

والسبب كما علمنا سابقا أن الأعداد التي يتكون منها نظام الستة عشر هي:

F, E, D, C, B, A, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

وحيث إن الحرف B هو في نطاق هذه الأعداد ، فقد تم إرجاع قيمة غير صفرية عن طريق الدالة isxdigit ، أما في حالة إدخال الحرف T فقد تم الرجوع بالقيمة صفر .

وهكذا يستمر تنفيذ البرنامج حتي إدخال الرقم 4 لإيقافه .

2) دوال تبديل الحرف *Character Alteration* **Functions**

توجد دوال أخرى أيضاً مهمتها تبديل الحرف من حرف كبير إلى حرف صغير وبالعكس .

I)) دالة تبديل الحرف إلى صغير : وهي الدالة tolower الناتجة من اختصار العبارة to lower case ومهمتها تبديل الحرف الكبير إلى حرف صغير ، وفي حالة كون الحرف كبيرا ترجع الدالة بعدد صحيح ويكون مقابلا لنفس الحرف في الشفرة ascii بشكله الصغير ومن ثم يجرى تبديله إلى حرف صغير ، أما إذا كان الحرف صغيراً فلا يتم تبديله .

II)) دالة تبديل الحرف إلى كبير : وهي الدالة toupper التي جآءت من اختصار العبارة to upper case ومهمتها تبديل الحرف الصغير إلى حرف كبير ، وفي حالة الحرف الصغير ترجع الدالة بعدد صحيح مقابل للحرف الكبير وبالتالي تبديله إلى حرف كبير ، وإلا فسيبقي الحرف كما هو .

مثال (2-7-7)

البرنامج يقوم بالتعرف على الحرف ، حيث يطبعه بصورته الصغيرة إذا كان الحرف المدخل حرفا كبيرا أو يبقي الحرف كما هو باستخدام الدالة tolower ، أيضاً استخدام الدالة toupper لتبديل الحرف ليصبح كبيرا إذا كان الحرف المدخل حرفا صغيرا وإلا فسيطبعه كما .

```
#include <conio.h>
#include <process.h>
#include <stdio.h>
#include <ctype.h>
   /* Program using tolower and toupper functions */
main()
{
```

```
        char ch;
        int answer, op;
        for( ; ; )
        {
          clrscr();
          printf("\nType 1 to use tolower function");
          printf("\n     2 to use toupper function");
          printf("\n     3 to exit ");
          printf("\n\nYour selection please : ");
          scanf("%d", &op);
          switch (op)
          {
                case 1 : printf("\nEnter character : ");
                         ch = getche();
                         printf("\nThe tolower case of ");
                         printf("%c = %c ", ch, tolower(ch) );
                         break;
                case 2 : printf("\nEnter character : ");
                         ch = getche();
                         printf("\nThe toupper case of ");
                         printf("%c = %c ", ch,toupper(ch) );
                         break;
                case 3:  exit(0);
          }
          getch();
        }
}
```

التنفيذ وإدخال الاختيار رقم 1 لاستخدام الدالة tolower كما يلي :-

```
Type 1 to use tolower function
     2 to use toupper function
     3 to exit
Your selection please : 1
```

وإدخال الحرف الكبير A

```
Enter character : A
```

سيكون الرد كالتالي :-

```
The tolower case of A = a
```

أما في حالة إدخال الحرف a

```
Enter character : a
```

ستكون الإجابة كالتالي :-

The tolower case of a = a

لاحظ أن الدالة tolower تقوم بتبديل الحرف إلى صورته الصغيرة إذا تحقق الشرط :

((ch>='A')&& (ch<='Z'))

أما إذا نفذ الاختيار رقم 2 أي تنفيذ الدالة toupper يتبعها الحرف A

Your selection please : 2

Enter character : A

فسيكون الرد :

The toupper case of A = A

أما في حالة إدخال الحرف a

Enter character : a

فعندها يتم استبداله بالحرف الكبير A كما يلي :-

The toupper case of a = A

8.7) تمرينات *Exercises*

1) اذكر الفرق بين دوال الادخال getch() ، getche() ، getchar مع توضيح ذلك ببعض الأمثلة .

2) ما هو الفرق بين :

a) printf() and puts()
b) atof and atoi
c) ispunct and isprint

3) أي من التعابير الآتية تعطي نتيجة (1) صواب :

a) "MARK FOX" == "MARK BOX"
b) "street no 234" < "street no 432"
c) "all done" != "all go"

4) اكتب برنامجا لقراءة اسمك بالكامل باستخدام دالة gets() ودالة getche() مع توضيح الفرق بينهما .

5) المطلوب كتابة برنامج يقوم بنسخ السلسلة S1 في السلسلة S2 .

6) اكتب برنامجا لحساب عدد الحروف B,G,N في سلسلة تم ادخالها عن طريق المفاتيح .

7) المطلوب كتابة برنامج يستقبل سلسلتين مع طباعة positive إذا كانت السلسلة الأولى أكبر من الثانية و negative إذا كانت الثانية أكبر من الأولى و zero إذا كانتا متطابقتين .

8) اكتب برنامجا لقراءة سلسلة حرفية حرفا حرفا ثم اطبع هذه الحروف بما يقابلها بالشفرة (ascii) مع إيقاف البرنامج إذا تم إدخال الرمز(?) .

9) اكتب برنامجا يقرأ عدد 5 جمل لا تتعدى الواحدة منها 20 حرفا ثم اطبع أقصر هذه الجمل .

10) اكتب برنامجا لقراءة سلسلة حرفية مع إيجاد مكان حرف معين بهذه السلسلة .

11) المطلوب كتابة برنامج مهمته استقبال سلسلة حرفية غير معروفة الطول ثم طباعتها عكسيا مع تحويل الحروف الصغيرة إلى كبيرة .

12) اكتب برنامجا لقراءة سلسلة حرفية لا يزيد طولها عن 50 حرفا ، حيث يكون من ضمنها الرمز (*) والمطلوب تحديد طولها ثم طباعة الحروف التي تسبق الرمز (*) في السطر الأول ، والحروف التي تلي الرمز (*) في السطر الثاني .

13) المطلوب كتابة برنامج يستقبل سلسلة حرفية ثم حساب :

* عدد الفراغات بالسلسلة .
* عدد الكلمات التي لا تحتوي على الحروف R, D .
* عدد الأرقام بالسلسلة .
* عدد تكرار الحرف K بالسلسلة .

14) اكتب برنامجا لإدخال عدد N من أرقام الهواتف ، المطلوب إضافة الرقمين 33 من الناحية اليسري للرقم الذي يبدأ بالرقم 3 ، وإضافة الرقمين 44 من الناحية اليسري للرقم الذي يبدأ بالرقم 4 .

15) المطلوب كتابة برنامج مهمته قراءة سلسلة حرفية لا يزيد طولها على 20 حرفا ثم اطبع هذه السلسلة عدد مرات طولها مع إلغاء الحرف الأخير من هذه السلسلة في كل مرة يتم فيها طباعتها وحتي النهاية ، فمثلا السلسلة STRING ستطبع بالشكل الآتي :-

```
STRING
STRIN
STRI
STR
ST
S
```

16) أوجد ناتج تنفيذ البرنامج الآتي :

```
#include <stdio.h>
main()
{
    char *str1 = "aaabbb",
         *str2 = "bbbccc",
         *str3 = "ccc";
    if( strncmp( str2, str1, 3) > 0 )
      printf("\n str2 > str1");
    else
      printf("\n str1 > str2");
    if( strncmp( str2, str3, 3) > 0 )
      printf("\n str2 > str3");
    else
      printf("\n str3 > str2");
}
```

17) اكتب برنامجاً كاملا مهمته قراءة قيم المتغير X ثم حساب قيمة المتغير Y حيث :

$$Y=f(X)+g(X)$$

$$f(X)=X^2+A$$

$$g(X)=\begin{cases} X^2+X & \text{if} \quad f(X)>0 \\ 2X+5 & \text{if} \quad f(X)\le 0 \end{cases}$$

على ان تستدعي دالة فرعية FX لحساب قيمة f(X) ودالة فرعية أخري GX لحساب قيمة g(X) ، المدخلات والمخرجات تكون بالدالة الرئيسية .

8

الفصل الثامن

مؤثرات الفاصلة والشرطي والبت

1.8) مؤثر الفاصلة The Comma Operator

إن مؤثر الفاصلة (,) الذى تتميز به لغة C يعتبر من أدنى المراتب بالنسبة لبقية المؤثرات عند التنفيذ ، ويستخدم هذا المؤثر في حالة وجود تعابير متعددة مفصولة عن بعضها بفواصل والقيمة النهائية تحسب من اليسار إلى اليمين ونوعها هو نوع التعبير بالطرف الأيمن.

الشكل العام

exp1 , exp2 , ... , expn ;

وهو عبارة عن عملية قد تحتوي على تعبيرين أو أكثر .

مثال (8-1-1)

خذ مثلا البرنامج الآتي :-

```
#include<stdio.h>
main()
{
    short a, b, j, k;
    j = 2;
    k = 15;
    j += 5, k--, j += k, k -= 1;
    printf("\nJ ==>%d   K ==>%d", j, k);
    a = 5;
    b = 4;
    b = a = 3*b-1, b += a;
    printf("\nA ==>%d  B ==>%d", a, b);
}
```

وفيه الجملة

```
j += 5, k--, j += k, k -= 1;
```

التي تحتوي على 4 تعابير يتم تنفيذها من اليسار إلى اليمين وهي :-

1) الأول j += 5 وينتج عنه تخصيص القيمة 7 للمتغير j .
2) الثاني --k وينتج عنه طرح القيمة 1 من قيمة المتغير k لتصبح 14 .
3) الثالث j += k وينتج عنه إضافة قيمة k الناتجة من الخطوة (2) إلى القيمة القديمة للمتغير j وهي 7 لتصبح قيمته تساوي 21 .
4) الرابع k -= 1 الذي ينتهي بالفاصلة المنقوطة (;) للدلالة على نهاية الجملة وهو يعني اطرح 1 من قيمة المتغير k وهي 14 لتصبح 13 .

يلي ذلك طباعة ناتج هذه المتغيرات التي تشابه الآتي :-

```
j ==>21   K ==>13
```

وبنفس الطريقة يتم تنفيذ الجملة

```
b = a = 3*b-1, b += a;
```

التي ينتج عنها الناتج الآتي :-

```
A ==>11   B ==>22
```

مثال (8-1-2)

اكتب برنامجا لحساب المعادلة الآتية :-

```
Z=(X-Y)/(X+Y)
```

حيث

X تساوي 1.0 , 1.5 , 2.0 , ... , 5.0

Y تساوي -5.0 , -4.5 , -4.0 , ... , -1.0

```
include<stdio.h>
main()
{
    float x, y, z;
    printf("  X      Y     Z\n\n");
    for(x=1 , y= -5 ; x <= 5.0 && y <= -1.0 ; x += 0.5 , y += 0.5)
    {
      if( x+y == 0 )
      {
        printf("Undefine Z at X=%.2f", x);
        printf(" and Y=%.2f\n", y);
      }
      else
      {
        z = (x-y)/(x+y);
        printf("%.2f   %.2f   %.2f\n", x, y, z);
      }
    }
}
```

بالبرنامج جرى استخدام مؤثر الفاصلة في جملة

for(x=1 , y=-5 ; x <= 5.0 && y <= -1.0 ; x += 0.5 , y += 0.5)

فالمؤثر الأول يعني تخصيص القيمتين 1 , 5- للمتغيرين y , x على التوالي وبالتالي إذا كانت قيمة التعبير (x+y) تساوي صفراً تغاض عن حساب قيمة z مع طباعة الرسالة المناسبة ، أما إذا كانت قيمة التعبير غير ذلك فاحسب قيمة المتغير z واطبع كلا من المتغيرات z, y, x وفي كلا الحالتين يتم تنفيذ مؤثر الفاصلة الثاني أي إضافة المقدار 0.5 لكل من المتغيرين y , x وهكذا يتم تكرار هذه الخطوات حتي تكون قيمة x أكبر من 5.0 وقيمة y أكبر من 1.0-.

البرنامج وقت التنفيذ سيعطي النتائج المشابهة للآتي :-

```
X       Y    Z
1.00  -5.00 -1.50
1.50 -4.50  -2.00
2.00 -4.00  -3.00
2.50 -3.50  -6.00
Undefine Z at X=3.00 and Y=-3.00
3.50 -2.50  6.00
4.00 -2.00  3.00
4.50 -1.50  2.00
5.00 -1.00  1.50
```

لاحظ أنه لم يجر طبع قيمة المتغيرات z, y, x عندما تكون قيمة x تساوي 3.0 وY تساوي 3.0- لأن التعبير (x+y) يساوي صفراً بل جرى طبع الرسالة التي تدل على أن قيمة z غير معرفة لدى تلكما القيمتين .

مثال (8-1-3)

لحساب مجموع الأعداد الفردية من 1 إلى 10 يمكن كتابة برنامج للحصول على المطلوب بالطريقة المعتاده الآتية :-

```
#include<stdio.h>
main()
{
      short i, sum = 0;
      for(i = 1 ; i < 10 ; i += 2)
      sum += i;
      printf("The sum is %d", sum);
}
```

هذا البرنامج يعطى الناتج الآتي :-

```
The sum is 25
```

مثال (8-1-4)

يمكن الحصول على المطلوب بالمثال السابق باستخدام مؤثر الفاصلة .

```
#include<stdio.h>
main()
{
    short i, sum;
    for(sum = 0, i = 1 ; i < 10 ; sum += i , i  += 2)
     ;
    printf("The sum is %d", sum);
}
```

حيث جرى استخدام مؤثر الفاصلة داخل جملة for وكذلك وضع الفاصلة المنقوطة (;) التي تدل على أنها جملة فارغة بعد جملة for مباشرة .

2.8) المؤثر الشرطي *The Conditional* Operator

ميزة أخرى تُضَافُ إلى لغة C وهي وجود المؤثر الشرطي الذي يرمز له بعلامة الاستفهام (?) وعادة ما يستخدم في المقارنة بين قيمتين وهو يأخذ الصيغة العامة :-

```
exp1 ? exp2 : exp3 ;
```

ويتكون من ثلاثة تعابير حيث يقوم المؤثر (?) بتقييم التعبير الأول exp1 فإذا كانت قيمته صحيحة أي لا تساوي صفرا ، عندها تحسب قيمة التعبير الثاني exp2 وبالتالي تكون هذه القيمة هي قيمة التعبير النهائية ، أما إذا كانت قيمة التعبير exp1 خاطئة أي تساوي صفرا فتحسب قيمة التعبير الثالث exp3 وتكون هذه القيمة هي قيمة التعبير النهائية .

مثال (8-2-1)

انظر إلى البرنامج الآتي .

```
#include<stdio.h>
main()
{
    int X,Y;
    X = 20;
    Y = X > 10  ?  200  :  100;
    printf("Y=%d", Y);
}
```

وفيه يتم تقييم التعبير الأول X>10 ونتيجته صحيحة أي أن قيمة المتغير X أكبر من 10 عليه تخصص قيمة التعبير الثاني الذي يلي المؤثر (?) وهي 200 إلى المتغير Y وتهمل قيمة التعبير الثالث الذي يلي الشارحة (:) وهي 100 .

عند التنفيذ يكون الناتج هو :

```
Y=200
```

مثال (8-2-2)

الجزء الآتي من البرنامج

```
if( a > b )
     printf("A=%d\n", a);
else
     printf("B=%d\n", b);
```

يقابله الآتي :-

```
a>b  ?  printf("A=%d", a) : printf("B=%d", b);
```

وهو يعني إذا كانت قيمة التعبير a>b صحيحة اطبع قيمة a أما إذا كانت غير ذلك اطبع b ، من هنا يمكن استخدام مؤثر الشرط (?) عوضا عن جملة (if-else) وكذلك العكس .

مثال (8-2-3)

البرنامج الآتي يقرأ عدد n من القيم الصحيحة ثم يحسب ويطبع عدد ومجموع كل من القيم الفردية والقيم الزوجية .

```
#include<stdio.h>
main()
{
    int i, n, odd, sum_odd, sum_even, even, value;
    even = odd = sum_odd = sum_even = 0;
    printf("\nType number of values: ");
    scanf("%d", &n);
    printf("Please enter %d values: ", n);
    for(i = 1 ; i <= n ; i++)
    {
       scanf("%d", &value);
       value % 2 !=0 ? odd++, sum_odd+=value :
                      (even++, sum_even+=value);
     }
    printf("\nWe have %d odd numbers ", odd);
    printf("and their sum is %d ", sum_odd);
    printf("\nWe have %d even numbers ", even);
    printf("and their sum is %d ", sum_even);
}
```

إذا نفذ هذا البرنامج وتم إدخال عدد القيم المطلوب إدخالها ولتكن 10 كما يلي :-

```
Type number of values: 10
```

ويتبعها إدخال القيم العشر كما يلي :-

```
Please enter 10 values: 2  7  10  12  6  11  8  12  5  10
```

عندها يكون رد البرنامج مشابهاً للآتي :-

```
We have 3 odd numbers and their sum is 23
We have 7 even numbers and their sum is 60
```

في هذا البرنامج استُخدِمَ المؤثر الشرطي

```
value % 2 !=0 ? odd++, sum_odd += value : (even++, sum_even+=value);
```

وفيه يتم تقييم التعبير الأول value % 2!= 0 فإذا كان ناتجه صحيحا ، فإن ذلك يعني أن القيمة المدخلة value هي قيمة فردية وعليه يتم تنفيذ التعبير الثاني الذي يضم جملتين وهما :-

الأولى odd++ تستعمل كعداد للقيم الفردية .

الثانية sum_odd += value تستعمل لحفظ مجموع القيم الفردية .

أما في حالة ما إذا كانت قيمة التعبير value % 2 != 0 غير صحيحة وهذا يعني أن قيمة value زوجية حينها يُجْرَى تنفيذ الجمل المحصورة بين الأقواس لكي يتم إحصاء عدد ومجموع القيم الزوجية .

3.8) الأنظمة العددية Numbers System

قبل الدخول في مناقشة مؤثرات التعامل مع البت ، ينبغي التطرق إلى الأنظمة العددية التي يتعامل معها الحاسب الآلي وهي :-

1) النظام العشري Decimal System

فيه يستخدم الأرقام :-

9, 8, 7, 6, 5, 4, 3, 2, 1, 0

وأساس هذا النظام 10 أي عشرة أرقام ومن هذه الأرقام العشرة يمكن تكوين ترقيم في النظام العشري الذي نتعامل به في حياتنا اليومية .

2) النظام الثنائي Binary System

ويعتبر هذا النظام من أنسب الأنظمة العددية ملاءمة للاستخدامات والتطبيقات في مجال الحاسبات الآلية وهو يتكون من الرقمين 0 , 1 .

3) النظام الثماني *Octal System*

أساس هذا النظام هو الرقم 8 لأنه يتكون من ثمانية ارقام هي :

7, 6, 5, 4, 3, 2, 1, 0

4) النظام الستة عشرى *Hexadecimal System*

ويتكون من :

F, E, D, C, B, A, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

وسمي بهذا الاسم لأنه يتكون من ستة عشر رقما وأساسه 16 .

وفيما يلي جدول يبين العلاقة بين هذه الأنظمة الاربعة .

النظـــــــام			
السنة عشرى	الثماني	الثنائي	العشري
0	0	0000	0
1	1	0001	1
2	2	0010	2
3	3	0011	3
4	4	0100	4
5	5	0101	5
6	6	0110	6
7	7	0111	7-
8	10	1000	8
9	11	1001	9
A	12	1010	10
B	13	1011	11
C	14	1100	12
D	15	1101	13
E	16	1110	14
F	17	1111	15

في حين الجدول الآتي يوضح بعض الأرقام العشرية الصحيحة ومايكافؤها من أرقام في الأنظمة العددية الأخرى .

النظـــــــــــام			
السنة عشرىَ	الثماني	الثنائي	العشري
80	178	10000000	128
5	5	00000101	5
F	17	00001111	15
5D	135	01011101	93
8	10	00001000	8
A5	248	10100101	165

ولأن معظم العناصر المستخدمة لحفظ البيانات في ذاكرة الحاسب هي ذات صفات تنائية مزدوجة ، لهذا استخدم النظام الثنائي لتمثيل البيانات داخل الحاسب الذي يضم رقمين مختلفين هما :-

1) الصفر (0) وهو يمثل حالة عدم وجود تيار كهربائي أي أن مفتاح الدائرة الكهربية مغلق تماما .
2) الواحد (1) وهو يمثل حالة وجود تيار كهربائي أي أن مفتاح الدائرة مفتوح تماما .

ومن هذا النظام جاءت كلمة بت (BIT) وهي اختصار للعبارة (Binary digit) التي تعتبر أصغر معلومة يمكن تخزينها داخل الذاكرة وهي تمثل إما الرقم 0 أو الرقم 1 ، اما بايت (byte) التي سوف نتطرق إليها فعن طريقها يتم تمثيل أي رقم أو رمز داخل الذاكرة وهي تتكون من ثمانية بتات (8 bits) .

4.8) مؤثرات البت Bitwise Operators

توجد أربعة مؤثرات مألوفة في لغة C مهمتها التعامل مع البت وهي :-

المؤثر	معناه
&	مؤثر الضرب and
|	مؤثر الجمع or
^	مؤثر الجمع المنفرد أو المقتصرة exclusive
~	مؤثر النفي not أو مؤثر مكمل الواحد

جميع المؤثرات السابقة هي ذات عنصرين ماعدا مؤثر النفي فهو ذو عنصر واحد وأولويات تنفيذهما من أعلى إلى أسفل حسب الاسبقية الموضحة بالجدول الآتي :-

المؤثر	التنفيذ
~	من اليسار إلى اليمين
&	من اليسار إلى اليمين
^	من اليسار إلى اليمين
|	من اليسار إلى اليمين

الجدول الآتي يبين عمل مؤثرات التعامل مع البت للمتغيرين Y,X .

X	Y	X&Y	X^Y	X|Y	~X
1	1	1	0	1	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

مثال (8-4-1)

على فرض أن المتغيرين a ,b يحملان القيمتين 93 ,165 على التوالي ، فالمطلوب استخدام مؤثرات التعامل مع البت عليهما .

قبل التطرق إلى استخدام هذه المؤثرات التي تتعامل مع الأعداد ذات النظام الثنائي (binary digits) الذي تعمل به اجهزة الحاسب ، يجب التنويه بأن الرقم 93 يكافئ 01011101 والرقم 165 يكافئ 10100101 بالنظام الثنائي ، وبالتالي يمكن إجراء العمليات على هذين الرقمين كما يلي :-

1) مؤثر (&)

* بالنظام الثنائي

```
a      = 0101    1101
b      = 1010    0101
-------------------------
a & b= 0000     0101
```

* بالنظام الستة عشرى

5D & A5 = 5

2) مؤثر (|)

* بالنظام الثنائي

```
a      = 0101    1101
b      = 1010    0101
-------------------------
a | b = 1111    1101
```

* بالنظام الستة عشرى

5D | A5 = FD

3) **مؤثر (^)**

* بالنظام الثنائي

```
a     = 0101   1101
b     = 1010   0101
--------------------------
a ^ b = 1111   1000
```

* بالنظام الستة عشرى

5D ^ A5 = F8

4) **مؤثر (~)**

* بالنظام الثنائي

```
 a = 0101  1101
---------------------
~a = 1010  0010
```

5.8) مؤثرات الإزاحة *Shift Operators*

بلغة C هناك مؤثران يستخدمان للإزاحة أو النقل وهما :-

المؤثر	معناه
>>	مؤثر الإزاحة إلى اليمين shift right
<<	مؤثرالإزاحة إلى اليسار shift left

ويأخذ هذا النوع من المؤثرات اما الصيغة :

variable = variable OP expression ;

أو الصيغة :

variable OP = expression;

حيث variable اسم المتغير و OP مؤثر الإزاحة إلى اليسار أو اليمين في حين أن expression التعبير الذي يحدد عدد الخانات التي ستزاح ويتم تنفيذ المؤثر (<<) ثم المؤثر (>>) من اليسار إلى اليمين .

مثال (8-5-1)

المطلوب إزاحة الحرف A الذى تعادل قيمته 65 بالنظام العشري مقدار خانتين إلى اليسار .

هنا إذا تم الإعلان عن المتغير letter من النوع الحرفي وإسناد الحرف A إليه على النحو الآتي :-

```
char  letter = 'A' ;
```

وعلى فرض أن عملية الإزاحة الآتية قد تمت حسب المطلوب بالشكل

```
letter = letter << 2;
```

أو بالشكل

```
letter << 2;
```

فإن قيمة الحرف A قبل الإزاحة هي 01000001 أي العدد 65 بالنظام العشري ستصبح بعد الإزاحة القيمة 00000100 أي القيمة 4 بالنظام العشري .

مثال (8-5-2)

نفس الحرف بالمثال السابق ولكن بإزاحة خانتين إلى اليمين .

```
char letter='A';
 letter >>= 2;
```

هنا ستكون قيمة الحرف A بعد الإزاحة هي 00010000 بالنظام الثنائي أو 16 بالنظام العشري .

مثال (8-5-3)

إذا كانت لدينا الإشهارات والتخصيصات للمتغيرات الآتية :-

```
int A, B, C, E;
char Z;
A = 10; B = 13; C = 93;
E = 'Z';
```

فالجدول الآتي يبين بعض عمليات الإزاحة على هذه المتغيرات :-

التعابير	النظـــــــــام			
	العشري	الثماني	الستة عشري	الثنائي
A	10	12	A	00001010
A <<=2	20	24	14	00010100
A>>=1	5	5	5	00000101
B	13	15	D	00001101
B >>=2	3	3	3	00000011
B <<=3	24	30	18	00011000
C	93	135	5D	01011101
C >>=4	5	\ 5	5	00000101
C <<=1	10	12	A	00001010
E	90	132	5A	01011010
E >>=5	2	2	2	00000010
E <<	16	20	10	00010000

مثال (8-5-4)

نقدم فيما يلي برنامجا كاملا مع التنفيذ استُخدِمت فيه جميع المؤثرات التي تتعامل مع بت .

```
main()
{
    int i, j, m, a, b, c;
    m = 128; a = 93; b = 165;
    printf("\n If we have the following declartion :");
    printf("\nint a=%d,b=%d,m=%d;\n", a, b, m);
    printf("\nThen the bitwise operators are :");
    printf("\n--------------------------------------\n");
    printf("\nExpression   Decimal   ");
    printf("Hexadecimal   Binary\n");
    printf("--------------------------------------\n");
    for(j = 1 ; j <= 6 ; j++)
    {
      switch (j)
      {
        case 1 : c = a; printf(" A\t\t%d", a);
```

```
              printf("\t   %02X\t      ", a);
              break;
     case 2 : c = b; printf("\n B\t\t%d", b);
              printf("\t   %02X      ", b);
              break;
     case 3 : c = ~b;printf ("\n ~B\t\t%d", ~b);
              printf("\t   %02X      ", ~b);
              break;
     case 4 : c = a & b; printf("\n A & B\t\t%d", c);
              printf("\t   %02X      ", c);
              break;
     case 5 : c = a | b; printf("\n A | B\t\t%d", c);
              printf("\t   %02X      ", c);
              break;
     case 6 : c = a ^ b; printf("\n A ^ B\t\t%d", c);
              printf("\t   %02X      ", c);
              break;
    }
    for(i = 0 ; i < 8 ; ++i)
     {
       m&c ? printf("1") : printf("0");
       c <<= 1;
     }
  }
}
```

وفيما يلي النتائج المشابهة عند تنفيذ هذا البرنامج :-

```
If we have the following declartion :
int a=93,      b=165,      m=128;

Then the bitwise operators are :
------------------------------------------------------------
Expression  Decimal  Hexadecimal  Binary
------------------------------------------------------------
   A          93          5D        01011101
   B          165         A5        10100101
   ~B         ~166        FF5A      01011010
   A&B        5           05        00000101
   A|B        253         FD        11111101
   A^B        248         F8        11111000
```

بعد الإعلان عن بعض المتغيرات وتخصيص قيم مناسبة لها وطباعة عدد من السطور التى تبين معلومات عن البيانات التي سوف يعالجها هذا البرنامج جاءت جملة for الخارجية التي مهمتها تكرار جملة switch عدد 6 مرات وذلك حتي يتم استخدام جميع المؤثرات التي تطرقنا إليها في هذا الفصل حيث تم الحصول على المخرجات بالنظام العشري عن طريق التوصيف (%d) والستة عشرى بالتوصيف (%x) أما بخصوص النظام الثنائي فقد تم الحصول عليه باستخدام جملة for الداخلية التي تضم الجملتين :-

```
m&c ? printf("1") : printf("0");
c<<=1;
```

حيث m لها القيمة 128 وهي تساوي بايت واحدة أي 10000000 بالنظام الثنائي و 0x80 بالنظام الستة عشرى أما المتغير c خُصِّصَت له قيمة المتغير a الذى له القيمة 93 التي تكافئ 01011101 بالنظام الثنائي فى الحالة الأولى عندما تكون قيمة المتغير j تساوي 1 .

وعليه ينفذ المؤثر الشرطي (?) الذي فيه التعبير (m&c) أى البت في أقصى اليسار لكل من المتغيرين c , m ويكون الناتج إما صحيحا وبالتالي يطبع 1 أو خاطئا فيطبع 0 ثم تتم عملية إزاحة واحدة للمتغير c يسارا فتصبح قيمته 10111010 وهكذا تتكرر عملية الطباعة والإزاحة حتي تنتهي جملة for ، وفيما يلي قيم المتغيرات i , m , c , m&c عند تنفيذ جملة for الداخلية :

I	m	c	m&c
0	1	0	0
1	1	1	1
2	1	0	0
3	1	1	1
4	1	1	1
5	1	1	1
6	1	0	0
7	1	1	1

مثال (8-5-5)

استنتج مهمة البرنامج الآتي .

```
#include <stdio.h>
main()
{
    int i, a, x, m = 128;
     printf("Type your number -->:");
    scanf("%d", &x);
    printf("Your number in binary = ");
    for(i = 1; i <= 8; I++)
    {
      putchar( m & x  ?  '1'  :  '0'  );
      x <<= 1;
    }
}
```

مثال (8-5-6)

البرنامج الآتي مهمته توضيح كيفية التعامل مع المؤثرات الخاصة بالإزاحة .

```
#include <stdio.h>
main()
{
    int k, i, j, m, n;
    char x, a, ch;
    x = 'A'; ch = 'X';
    m = 128; n = 5;
    printf("\nIf we have the following declarion");
    printf("\nchar x='%c',ch='%c';int m=%d;\n", x, ch, m);
    printf("\nThen the shift operators are :");
    printf("\nExpression    Decimal   ");
    printf("Action       Binary\n");
    printf("----------------------------------\n");
    for(k = 1 ; k <= 2 ; k++)
    {
      for(j = 1 ; j <= 3 ; j++)
      {
```

```
switch (j)
{
  case 1 : a = x;
          printf("\n   %c\t\t", ch);
          printf("%d\tunshifted   ", x);
          break;
  case 2 : a = x<<n;
          printf("\n%c<<=%d\t\t", ch, n);
          printf("%d\tleft  %d\t   ", a, n);
          break;
  case 3 : a = x>>n;
          printf("\n%c>>=%d\t\t", ch, n);
          printf("%d\tright %d    ", a, n);
          break;
}
for(i = 0 ; i < 8 ; ++i)  /* Print value of a in binary  */
{
  putchar( m&a  ?  '1'  :  '0' );
  a <<= 1;
}
n -= 2;
}
printf("--------------------------------------");
x = 'Z';    /*  let x=Z  */
ch = 'Y';   /*  let ch=Y */
n = 7;      /*  let n=7  */
}
}
```

إذا نفذ هذا البرنامج ، فسيكون الناتج مشابها للآتي :-

```
If we have the following declarion
char x='A',ch='X';int m=128;

Then the shift operators are :
Expression  Decimal  Action     Binary
-----------------------------------------------
   X          65   unshifted   01000001
   X<<=3      8    left 3      00001000
   X>>=1      32   right 1     00100000
   Y          90   unshifted   01011010
   Y<<=5      64   left 5      01000000
   Y>>=3      11   right 3     00001011
```

وفيه جملة for الخارجية وتقوم بالمهمتين التاليتين مرتين وهما :-

أولاً : جملة for التي تقوم بتكرار عدد من الجمل ثلاث مرات وهي :

1) جملة switch التي تضم ثلاثة حالات لإزاحة قيمة المتغير x وبالتالي إسناد عملية الإزاحة في كل حالة للمتغير a .
2) جملة for الأخيرة ومهمتها طباعة قيمة المتغير a بالنظام الثنائي مع إزاحته مرة واحدة لليسار لعدد 8 مرات .
3) إنقاص قيمة n بمقدار 2 لإجل إزاحة قيمة المتغير z في المرة الموالية.

ثانياً : تنفيذ دالة printf () لغرض رسم خط يفصل المخرجات السابقة عن اللاحقة. وإسناد الحرفين Y , Z للمتغيرين ch , x على التوالي والقيمة 7 للمتغير n حتي يتم إجراء الخطوات (2,1) السابقة لهذه المتغيرات .

6.8) تمرينات Exercises

1) على فرض أن لدينا الجزء الآتي من البرنامج

```
int n = 8;
n = n % 5;
if( n == 1 )
  printf("one");
else
  if( n == 2 )
      printf("two");
      else
      printf("three");
```

* ما هو الناتج ؟
* أعد كتابة السابق باستخدام المؤثر الشرطي (?) .
* هل المؤثر الشرطي (?) يحل محل جملة if-else دائما؟

2) أذا أُعْطيتَ الجمل :

```
int result, i = 2, j = 3, k = 11;
char x = 'a', y = 'b', z = 'c';
```

ما هي قيمة المتغير result من خلال الفقرات الآتية :-

```
a)   result = i % j == i ? k : j ;
b)   result = i*j == k-(j+i) ? x : k ;
c)   result = y-1 != x ? k : j ;
d)   result = --z == y++ ? i*k : i*j ;
e)   result = z >= x+i ? z : x ;
f)   result = k % (i+j) == 1  ? k>>1 : k>>1 ;
```

3) أعد كتابة الفقرات (f, d) بالتمرين (2) باستخدام جملة if فقط .

4) المطلوب كتابة برنامج يقوم باستقبال M من الأعداد الصحيحة الموجبة ثم تحويلها إلى النظام الثنائي والثماني والستة عشرى .

5) إذا تم إشهار المتغيرين b, a من النوع الصحيح وخُصِّصَت لهما القيم 79, 90 على التوالي ، أوجد قيم كل من الفقرات الآتية :-

a) a & b;
b) a | b;
c) a ^ b;
d) a << 1;
e) b >> 1;

6) اكتب برنامجا مهمته تحويل رقم صحيح إلى ما يقابله في النظام الثنائي ثم يعكس الناتج ، فمثلا الرقم 43 يعطى البايت 00101011 في حين 11010100 هو الناتج النهائي .

7) أوجد ناتج تنفيذ البرامج الآتية :-

a)

```
#include <stdio.h>
main()
{
    char ch1, ch2 = 'A';
    printf("\nThe result ==> %c & %c",
          ( ch1 = ch2, ch1='B'));
}
```

b)

```
#include <stdio.h>
main()
{
    int a = 9, b = 4;
    printf("\na = %d  b= %d  c= %d", a, a>>1 , a>>2);
    printf("\nd = %d  e= %d  f = %d", B, B<<1 , B<<2);
}
```

c)

```
#include <stdio.h>
main()
{
    int m = 0xe, n = 2;
    printf("%x\n", m | (1<<n));
    printf("%x\n", m & (1<<n));
    printf("%x\n", (n^(1<<n)) & (m^0));
}
```

9

الفصل التاسع

الــــدوال والمؤشرات

1.9) تعريف الدالة Function Definition

كُتِبت البرامج في الفصول السابقة على أنها كتلة واحدة غير مجزأة ، وهذا قد يكون غير مناسب في بعض الأحوال حيث يصعب متابعتها وإصلاحها خصوصاً عندما يكون البرنامج طويلا .

لذلك فإنه من الأفضل تجزئة البرنامج الواحد إلى عدة أجزاء صغيرة كل جزء منها يؤدي مهمة معينة بحيث يتم اختبار هذه الأجزاء ومن ثم ربطها مع بعضها لِتُكَوِّنَ البرنامج الكامل ويمكننا عمل ذلك باستخدام الدالة .

وهناك وجه آخر لاستخدام الدالة لأنه في أحيان كثيرة يتحتم على المبرمج إعادة تنفيذ عدد من الجمل المتكررة في البرنامج الواحد ، وفي هذه الحالة يمكننا تحويل هذه الجمل المتكررة إلى دوال يمكن استدعاؤها عن طريق اسم الدالة .

الشكل العام للدالة

```
type  function_name (par1, par2, ...)
types  of the parameter list variables;
{
        types  of local variables ;
        function body ;
        return (expression) ;
}
```

حيث :

type نوع قيمة الدالة عند رجوعها إلى البرنامج المنادي .

function_name يمثل اسم الدالة .

par2, par1 معاملات أو دلائل لاستقبال وإرجاع البيانات .

types of the parameter نوع المعاملات المستقبلة والمرجعة للبيانات .

types of local variable المتغيرات المعلن عنها في الدالة وهي محلية أي تستخدم في نطاق الدالة فقط .

body جسم الدالة قد يكون جملة أو مجموعة جمل داخل القوسين { } .

return وهي إرجاع قيمة التعبير إلى مكان استدعاء الدالة .

2.9) ملاحظات عن الدالة Remarkes on Function

وقت استخدام الدالة يجب مراعاة الآتي :-

1) من الضروري أن يتبع القوسان ( ) اسم الدالة، خذ مثلا هذه الدالة :

```
sample()
{
    .
    .
    .
}
```

وقد يحتوي القوسان على معاملات (Parameters) وقت تبادل البيانات بين البرنامج الرئيسي والدالة ، أو قد لا يحتويان على شيء .

2) اسم الدالة لا يتبعها الفاصلة المنقوطة (;) بعد القوسين ( ) ، ومثال ذلك:-

```
funy(value1 , value2)
```

أما عند اسـتدعاء الدالـة فيجـب أن تنتهـي بالفاصلـة المنقوطـة ، أي يمكـن استدعاء الدالة funy كما يلي :-

```
var1 = funy (value1 , value2) ;
```

3) ينبغي الإعلان عن نوع الدالة إذا كانت ترجع بقيمـة مـن النـوع غير الصحيح int وفي حالة عدم الإعلان عنها قبل استدعائها تكـون قيمتهـا صحيحة int تلقائيا ، ولكن يستحسن الإعلان عنهـا فـي كـل الأحـوال ، فالدالتان التاليتان تعطيان قيمة من النوع الصحيح int .

```
int example( )          example ( )
{                       {
      .                       .
      .                       .
      .                       .
}                        }
```

4) قد تأتي الدالة قبل أو بعد البرنامج الرئيسي وعادة ما تأتي بعده .

3.9) الدالة الفارغة *void function*

الدالة الفارغة هي التي تكون لها مهمـة معينـة تؤديهـا بـدون إرجـاع قيمـة عند انتهائها أي أنها دالة خالية بـدون معـاملات (Parameters) ويكـون شـكلها عند تعريفها :

```
void nothing(void)
```

أما عند استدعائها تكون كالآتي :-

```
nothing();
```

قد يتم استخدام العينة (Prototype) حتي يمكن التغلب على اختلاف البيانـات المرسلة إلى الدالة ، فالإعلان الآتي :-

```
float   find_max_number (num1, num2);
```

يعني أن الدالة find_max_number تقوم بتحويل كل القيم المرسلة اليها عن طريق المعاملات (Parameters) الحقيقية num1, num2 إلى قيم من النوع الحقيقي مع الرجوع بقيمة حقيقية ، وبالتالي يمكن مناداة الدالة على النحو الآتي :-

```
find_max_number (12.0 , 5.0);
```

حيث كل المعاملات قيم حقيقية float .

أو على النحو الآتي :-

```
find_max_number (12 , 5);
```

حيث المعاملات قيم صحيحة int .

4.9) جملة return

هذه الجملة قد تأخذ الشكل الآتي :-

```
return(expression);
```

وعند الوصول إليها يتم تقييم التعبير expression والرجوع بهذه القيمة حسب نوع الدالة إلى مكان استدعائها ، وقد لا تستعمل عند عدم الرجوع بأي قيمة ، وقد تأخذ بعض الأشكال الأخرى مثل :

```
return ( a+b );          return ( ++a );          return sum/n;
```

مثال (9-4-1)

البرنامج الآتي يوضح استخدام الدالة void() .

```
#include <stdio.h>
main()
{
     void stars(void);  /*  Prototype the function  */
     void line(void);  /*  Prototype the function  */
     stars();             /* Call function star */
     line();              /* Call function line  */
     stars();             /* Call function star */
}
void line(void)
{
    printf("\n* Welcome to C book *");
}
void stars(void)
{
    printf("\n********************");
}
```

الناتج من هذا البرنامج عند تنفيذه هو :

```
********************
* Welcome to C book *
********************
```

الذي نلاحظه في الدالة الرئيسية ()main هو ما يلي :-

1) تم الإعلان عن نوع الدالة باستخدام العينة Prototype .
2) تم استدعاء الدالة stars أولا ثم الدالة line ثانيا واستدعاء الدالة stars ثالثا .
3) انتهاء الدالة بالفاصلة المنقوطة عند تعريفها وعند استدعائها .

أما بخصوص الدالتين line, stars فيجب مراعاة ما يلي :-

1) يجب أن تكونا متوافقتين من حيث النوع والاسم المعلن عنهما في الدالة الرئيسية .
2) اسم الدالة يتبعه القوسان () بداخلها void أي الدالة خالية من أي قيمة ومن غير أن تنتهي بالفاصلة المنقوطة .

3) جسم الدالة يجب أن يكون محصوراً بين القوسين { } حيث تمت طباعة سلسلة من النجوم عند استدعاء الدالة stars يليها طباعة السطر :

Welcome to C book

ثم طباعة النجوم مرة ثانية .

4) عدم نهاية أي من الدالتين بالأمر return لعدم الرجوع بأي قيمة .

5.9) المتغيرات المحلية *Local Variables*

وهي متغيرات يتم الإعلان عنها في حدود الدالة ولا يمكن التعرف عليها في الدالة الرئيسية أو أية دالة أخرى حتى ولو كانت تحمل نفس الاسم .

مثال (9-5-1)

البرنامج الآتي يستخدم أكثر من دالة بدون معاملات وذلك لإجراء عمليات الضرب والقسمة على قيمتين .

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
main()
{
    int op;
    void display();
    void mul();
    void div();
    do
    {
      clrscr();
      display();  /* Call function for selection list  */
      scanf("%d", &op);
      switch (op)
      {
        case 1 : mul();  break;
        case 2 : div();  break;
        case 3 : exit(0);
```

```
        }
        getch();
    } while( (op != 1) || (op != 2) );
    printf("\n***** GOOD BYE *****");
    return 0;
}

void display()      /* Display function */
{
    printf("Type   1. Multiplication  \n");
    printf("         2. Division        \n");
    printf("         3. Exit            \n");
    printf("Enter your selection : ");
}
        /* Multiplaction function */
void mul()
{
    float n1, n2;
    double m;
    printf("\nPlease give two numbers : ");
    scanf("%f %f", &n1, &n2);
    m = n1*n2;
    printf("NOW %.2f * %.2f = %.2f\n", n1, n2, m);
}

void div()      /* Division function */
{
    float x, y, d;
    printf("\nPlease give two numbers : ");
    scanf("%f %f", &x, &y);
    if( y == 0 )
    {
      printf("Floating point error:");
      printf(" Divide by zero.\n");
    }
    else
    {
      d = x/y;
      printf("NOW %.2f / %.2f", x, y);
      printf(" = %.2f\n", d);
    }
}
```

البرنامج تم فيه إشهار عدد ثلاث دوال فرعية وهي :-

1) الدالة display() ومهمتها عرض كل الاختيارات المتاحة أمام المستخدم على شاشة العرض .

2) الدالة mul() ووظيفتها استقبال قيمتين ثم طباعة حاصل ضربهما .

3) الدالة div() ومهمتها استقبال قيمتين ثم طباعة حاصل قسمتيهما في حالة كانت القيمة الثانية لا تساوي صفرا وإلا فستُطْبَعُ الرسالة المناسبة .

عند التنفيذ يتم مسح شاشة العرض عن طريق الأمر clrscr() تأتي بعدها قائمة الاختيارات مع الرسالة

Enter your selection :

وعليه لإجراء عملية القسمة مثلا ، عليك بإدخال الرقم 1 ليتم إسناده للمتغير op وعن طريق جملة switch يتم استدعاء الدالة ;div() التي هي بدون معاملات حيث تم الإعلان عن المتغيرات d, y, x من النوع الحقيقي وتعتبر هذه المتغيرات جميعها محلية أي تخص هذه الدالة فقط ، يلي ذلك إدخال قيمتين من النوع الحقيقي ومن ثم تنفيذ عملية القسمة وطباعة الناتج في حالة كون قيمة y لا تساوي صفراً أما إذا كان غير ذلك فستطبع الرسالة :

Floating point error : Divide by zero.

وبالضغط على أي مفتاح يرجعك البرنامج إلى قائمة الاختيارات وهكذا تتكرر العملية حتي يتم الخروج نهائيا من البرنامج بالضغط على أي رقم عدا الرقمين 2, 1 .

الملاحظ في كل الدوال السابقة أنه لم يعلن عن أنواعها لأنه لا حاجة لذلك بسبب أنها بدون معاملات ، أي لا يمكن تمرير قيم من الدالة الرئيسية إلى الدالة الفرعية وبالعكس .

وأخيرا هذه بعض النتائج عند تنفيذ هذا البرنامج .

```
Type  1. Multiplication
      2. Division
      3. Exit
Enter your selection : 1
Please give two numbers : 3.5  2.5
NOW 3.50 * 2.500 = 8.75

Type  1. Multiplication
      2. Division
      3. Exit
Enter your selection : 2
Please give two numbers : 19   4
NOW 10.00 / 4.00 = 4.75

Type  1. Multiplication
      2. Division
      3. Exit
Enter your selection : 3
*****  GOOD BYE *****
```

مثال (9-5-2)

اكتب برنامجا رئيسياً لقراءة عدد أربعة متغيرات حقيقية ثم استخدم الدالة لإيجاد أكبر قيمة من هذه المتغيرات .

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
main()
{
    float temp1, temp2, max;
    float num1, num2, num3, num4;
    void nothing();                  /* Prototype the function  */
    float larger(float , float);     /* Prototype the function  */
    nothing();                       /* call nothing function  */
    scanf("%f %f", &num1, &num2);
    scanf("%f %f", &num3, &num4);
    temp1 = larger(num1, num2);      /* call larger function */
    printf("\nThe larger of the first ");
    printf("two items is %.3f", temp1);
    temp2 = larger(num3, num4);      /* call larger function */
```

```
    printf("\nThe larger of the second ");
    printf("two items is %.3f", temp2);
    max = larger(temp1, temp2);       /* call larger function */
    printf("\nThe largest number");
    printf(" is %.3f", max);
    getch();
    return 0;
}

void nothing()
{
    printf("\n Find the largest of four numbers ");
    printf("\n -------------------------------------");
    printf("\n Enter four data items: ");
}

float  larger(float x, float y)
{
   float ansr;
    ansr = x>y ? x : y;
    return (ansr);
}
```

ما سيطبعه البرنامج عند تنفيذه هو الآتي :-

```
Find the largest of four numbers
-------------------------------------
Enter four data items: 9.999 5.555 0.044 0.444

The larger of the first two items is 9.999
The larger of the second two items is 0.444
The largest number is 9.999
```

شرح البرنامج :

بدأت الدالة الرئيسية بالإعلان عن نوع الدالة

```
void nothing();
```

وتعني أنها دالة بلا معاملات كما تم شرحها سابقا ، ثم الإعلان عن الدالة الثانية

```
float larger(float , float);
```

وتعني أن لها معاملين حقيقيين وسترجع بقيمة من النوع الحقيقي .

جاء بعد ذلك استدعاء الدالة الخالية nothing() التي مهمتها طباعة الآتي:-

```
Find the largest of four numbers
----------------------------------------
Enter four items:
```

ثم الرجوع إلـى البرنـامج الرئيسـي لإدخـال أربـع قيـم حقيقيـة ، يلـي ذلـك استدعاء الدالة larger() عن طريق الجملة :

```
temp1 = larger(num1, num2);
```

حيث تمرر قيمة كل من المعاملات num2, num1 إلى العنصرين y, x فـي الدالـة larger() ومـن ثـم تجـرى مقارنـة هذيـن العنصريـن والرجـوع بالقيمـة الكبرى وتخصيصها للمتغير temp1 بالدالـة الرئيسـية وهكذا يتـم نفس الشـئ كلما استدعيت الدالة ولكن بقيم مختلفة .

أما فيما يخص شكل الدالة:

فقد بدأت بالإعلان عن نوعها كمايلي :-

```
float  larger(float x, float y)
```

وهي تدل على أنها دالة من النوع الحقيقي مطابقة من حيث النوع والاسـم لما هو موجود بالدالة الرئيسية وتحتوي على معاملين y, x حيث اعلـن عـن هذين المتغيرين من النوع الحقيقي داخل القوسين() الخاصين باسم الدالة ، أمـا جسم الدالة فيحتوي على المؤشر الشرطي (?) الذي مهمتـه المقارنـة بين قيم المتغيرين y, x والرجوع بأكبرهما بواسطة المتغير المحلي ansr.

نلاحظ في استخدام الدالة ميزة كبـيرة.وهـي تجنـب تكـرار جملـة المقارنـة لأكثر من مرة في الدالة الرئيسية .

مثال (9-5-3)

ماذا يحدث إذا تغير الإعلان عن نوع الدالة larger() بالمثال السابق أي كالآتي :-

```
int  larger(float x, float y)
{
   float ansr;
    ansr = x>y ? x : y;
    return (ansr);
}
```

الجواب :-

عند تنفيذ البرنامج وإدخال نفس البيانات بالمثال السابق سيكون الناتج مشابها لما يلي :-

```
Find the largest of four numbers
-----------------------------------------
Enter four data items: 9.999 5.555 0.044 0.444
The larger of the first two items is 9.000
The larger of the second two items is 0.000
The largest number is 9.000
```

أول ما نلاحظه هنا عدم توافق هذه النتائج مع مثيلتها في المثال السابق ، والسبب هو أنه تم الإعلان عن الدالة من النوع الصحيح int وبالتالي تمت طباعة كل القيم الصحيحة فقط دون الأعداد الكسرية ، وهذا خطأ شائع عند استخدام الدوال .

عند تمرير البيانات من البرنامج الرئيسي إلى الدالة يجب أن تكون عدد المعاملات المتصلة مع بعضها البعض متوافقة من حيث العدد والنوع .

مثال (9-5-4)

انظر إلى البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    float a;
    float squre_number(int);  /*  Prototype the function  */
    printf("\nEnter value of x ==> ");
    scanf("%f", &a);
    printf("\nThe squre of %.2f", a);
    printf(" is %d", squre_number(a) );
}

float squre_number(int x)
{
    return (x*x);
}
```

هذا البرنامج يوضح العلاقة بين الدالة الرئيسية والدالة الفرعية من حيث نوع المعاملات ، فعند استدعاء الدالة تمرر قيمة المتغير الحقيقي a إلى المعامل المقابل لها في الدالة وهو المتغير الصحيح x ، ومن هنا نلاحظ الاختلاف بين هذين المتغيرين وعليه تخزن قيمة 0 في المتغير x وبالتالي الرجوع بمربع هذه القيمة إلى نقطة الاستدعاء حيث يتم طباعتها هناك ، والناتج يكون :

```
Enter value of x ==> 5
The squre of 5.00 is 0
```

وهي نتيجة خاطئة بطبيعة الحال لأن تربيع العدد 5 هو 25 وليس 0 .

إذاً ما هو الحل في هذه الحالة ؟

الجواب

في لغة C يمكن حل هذه المعضلة بالإعلان عن نوع الدالة قبل استدعائها كما يوضحه البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    float a;
    int squre_number(int); /*  Prototype the function  */
    printf("\nEnter value of x ==> ");
    scanf("%f", &a);
    printf("\nThe squre of %.2f", a);
    printf(" is %d", squre_number(a) );
}

int  squre_number(int x)
{
    return (x*x);
}
```

بعد الإعلان عن المتغير a من النوع الحقيقي جاء الإعلان

```
int squre_number(int);
```

عن نوع الدالة قبل استدعائها وهي تعني أنها دالة من النوع الصحيح int ولها معامل واحد (Parameter) من النوع الصحيح أيضاً حيث وضع نوع القيمة المرسلة int بين القوسين في نهاية تعريف الدالة وهو ما يعرف بالعينة (Prototype) التي سبق شرحها ، عند تنفيذ هذا البرنامج سوف يولد الناتج الآتي :-

```
Enter value of x ==> 5
 The squre of 5.00 is 25
```

مثال (5-5-9)

المطلوب كتابة برنامج لحساب متوسط N من القيم الصحيحة على أن يتم حساب المتوسط عن طريق دالة فرعية .

```
#include <stdio.h>
main()
{
    int n;
    float averg, naverage(int);
    printf("\nEnter number of values ==> ");
    scanf("%d", &n);
    averg = naverage(n);
    printf("The average of all values is %.3f", averg);
}
float naverage(int a)
{
    int i, value;
    float sum = 0.0;
    printf("Enter %d values please ==> ", a);
    for(i = 1 ; i <= a ; i++)
    {
      scanf("%d", &value);
      sum += value;
    }
    return sum/a;
}
```

بالبرنامج تم الإعلان عن اسم الدالة naverage وتعني أن معاملها من النوع الصحيح وأنها ترجع بقيمة حقيقية ، أما فيما يخص الدالة :

1) فهي تستقبل قيمة صحيحة عن طريق المتغير n يتم تخزينها في المتغير a .

2) المتغيرات sum, value, i تعتبر متغيرات محلية تستخدم في نطاق هذه الدالة فقط .

3) الرجوع بقيمة حقيقية واحدة ناتجة من الجملة return .

وهذا هو ناتج تنفيذ هذا البرنامج

```
Enter number of values ==> 5
Enter 5 values please ==> 9  2  7  10  6
The average of all values is 6.800
```

مثال (9-5-6)

المطلوب كتابة برنامج مهمته حساب مضروب عدد من النوع الصحيح .

الحل

هذا البرنامج يمكن كتابته بعدد قليل من الجمل، ولكي نوضح كيفية استدعاء دالة لدالة أخرى تمت تجزئته إلى مجموعة من الدوال الفرعية يتم استدعاؤها عن طريق الدالة الرئيسية main() وهذه الدوال هي :-

1) الدالة الاولى read_n() تستخدم للحصول على العدد المطلوب حساب مضروبه ثم فحص العدد ، فإذا كان سالبا تستدعي الدالة abs_n وإلا فيتم استدعاء الدالة information .

```
void read_n()
{
    int n;
    int test(int p); /*  Prototype the function  */
    int abs_n(int a); /*  Prototype the function  */
    printf("\nPlease enter n ==> ");
    scanf("%d", &n);
     test(n)  ? abs_n(n) : information(n);
}
```

2) الدالة الثانية test(int p) وهي لاختبار ما إذا كانت قيمة المتغير p موجبة أو سالبه :

```
int test(int p)
{
    int ansr;
    ansr = p <= 0 ? 1 : 0;
    return ansr;
}
```

3) الدالة الثالثة abs_n(int a) لإيجاد الحد المطلق للمتغير a.

```
int abs_n(int a)
{
    information ( abs(a) );
}
```

4) الدالة int information(int m) لكتابة الرسالة لإدخال العدد المطلوب إيجاد مضروبه واستدعاء الدالة calculate التي تحسب مضروب العدد .

```
int information(int m)
{
    int calculate_n(int); /*  Prototype the function  */
    printf("\nThe factorial of %d = ", m);
    calculate_n(m);
}
```

5) الدالـة الخامسـة calculate_n(int x) مهمتهـا الحصـول علـى مضـروب المتغير x واستدعاء الدالة print_n .

```
int calculate_n(int x)
{
    long print_n(long); /*  Prototype the function  */
    short k = 1;
    long fact = 1;
    while( k <= x )
    {
      fact *= k;
      ++k;
    }
    print_n(fact);
}
```

6) الدالـة الأخيـرة print_n(factorial) تسـتخدم لإظهـار النـاتج علـى شاشـة العرض لقيمة المتغير factorial .

```
long print_n(long factorial)
{
    printf("%ld", factorial);
}
```

وأخيرا تكتب الدالـة الرئيسـية main () لاسـتدعاء الـدوال السـابقة التـي قـد تكون كالآتي :-

```
#include <stdio.h>
#include <math.h>
int information(int m); /*  Prototype the function  */


main()
{
    void read_n();/*  Prototype the function  */
    read_n();
}
```

وعند تنفيذ هذا البرنامج وإدخال العدد 6 كالآتي :-

Please enter n ==> 6

سيظهر الناتج عن طريق السطر الآتي :-

The factorial of 6 = 720

6.9) المتغيرات الخارجية *External Variables*

تعرفنا سابقا على المتغيرات المحلية أو ما يعرف بالداخلية حيث يعلن عن المتغير داخل جسم كل دالة وبالتالي يتم إستخدامه في نطاق تلك الدالة ولا علاقة لها بالدوال الأخرى .

أما المتغيرات الخارجية أو العامة فهي التي تكون معروفة لجميع الدوال الموجودة في الدالة الرئيسية ، حيث يتم الإعلان عنها خارج كل الدوال أي قبل بداية الدالة الرئيسية main() .

مثال (9-6-1)

البرنامج الآتي يبين الفرق بين المتغير الداخلي والمتغير الخارجي .

```
#include <stdio.h>
int k = 5;              /* k is an external variable */
main()
{
    int call_it(int j);  /* Prototype the function */
    int j = 5;
    printf("\n\t THE OUT PUT ");
    call_it(j);
    printf("\nThe first call K=%d  J=%d", k, j);
    call_it(j);
    printf("\nThe second call K=%d  J=%d", k, j);
}
int call_it(int m)
{
    printf("\n-------------------------");
    k *= k;
    m *= m;
    return 0;
}
```

التنفيذ يولد الآتي :-

```
        THE OUT PUT
    ------------------------------
    The first call K=25  J=5
    The second call K=625  J=5
```

في هذا البرنامج تم الإعلان عن المتغير الخارجي k وإسناد القيمة 5 له قبل بداية الدالة الرئيسية main() ، في حين يعتبر المتغير j متغيرا داخليا إسندت القيمة 5 له وهو معروف للدالة الرئيسية فقط .

عند استدعاء الدالة call_it في المرة الأولى تم إرسال قيمة المتغير j وهي القيمة 5 إلى دليل الدالة m الذي يعتبر متغيرا محليا لهذه الدالة حيث تمت مضاعفة كلٍ من المتغيرين m, k ليصبح كل منهما مساويا للقيمة 25 ، عند الرجوع إلى نقطة الاستدعاء في الدالة الرئيسية ، لم تتغير قيمة المتغير j

فبقيت قيمته كما هي 5 في حين تمت طباعة قيمة المتغير الخارجي k وهي 25، وهكذا حدث نفس الشيء عند استدعاء الدالة في المرة الثانية .

ولكن ماذا يحدث إذا ما تم إعادة الإعلان عن المتغير الخارجي مرة أخرى؟ المثال الآتي يبين ذلك .

مثال (9-6-2)

يمكن إعادة كتابة الدالة الفرعية call_it بالبرنامج السابق حتى نلاحظ ما يحدث .

```
int call_it(int m)
{
    int k;
    printf("\n---------------------------");
    k *= k;
    m *= m;
    return 0;
}
```

نلاحظ في الدالة أنه قد تمت إعادة الإعلان عن المتغير k من النوع الصحيح وبالتالي يصبح متغيرا محليا خاصا لهذه الدالة مثله مثل المتغير المحلي m ولا يتم ارجاع قيمته إلى الدالة الرئيسية وعليه إذا ما نفذت هذه الدالة مع الدالة الرئيسية فسيكون الناتج هو :

```
THE OUT PUT
------------------------------
The first call K=5  J=5
The second call K=5  J=5
```

7.9) المتغيرات الساكنة Static Variables

هي متغيرات ساكنة لأنها تحتفظ بقيمتها مخزنة ما دام البرنامج باقيا في التنفيذ .

الشكل العام

```
static type variable;
```

مثال (9-7-1)

البرنامج الآتي يوضح استخدام المتغيرات الساكنة .

```
#include <stdio.h>
main()
{
    int i;
    int static_fun(int i);        /* Prototype the function */
    for(i = 1 ; i <= 5 ; i++)
      static_fun(i);
}

int static_fun(int b)
{
    static int c = 1;
    printf("\nB= %d  C=%d", b, c);
    c += b;
}
```

يولد هذا البرنامج النتائج الآتية :-

```
B=1     C=1
B=2     C=2
B=3     C=4
B=4     C=7
B=5     C=11
```

هنا مُرِّرت قيمة المتغير i من الدالة الرئيسية إلى المتغير المحلي b بالدالة static_fun وفي هذه الدالة أعلن عن المتغير c على أنه متغير صحيح ساكن وخصص له العدد 1 مع طباعة قيمة كل من المتغيرين c, b ثم أضيفت قيمة المتغير b إلى قيمة المتغير c وفي كل مرة تُسْتَدْعَى الدالة لتطبع قيمة المتغير الساكن c المحتفظ بها في المرة السابقة .

أما إذا حذفت static فسوف يطبع العدد 1 في كل مرة تستدعى فيها الدالة.

مثال (9-7-2)

اكتب برنامجا لقراءة عدد صحيح n ثم أوجد السلسلة المسماة fibonacci حيث كل عدد فيها يساوي حاصل مجموع العددين اللذين يسبقانه أي:-

1+1+2+3+5+8+13+21+ ... n

```
#include <stdio.h>
main()
{
      int n, i = 1, call_fun(int i);
      scanf("%d", &n);
      printf("The Fibonacci ==> %d", i);
      for(i = 2; i < n; i++)
      {
         int ans;
         ans = call_fun(i);
         printf(" + %d", ans);
      }
}

int call_fun(int b)
{
    static int f1 = 1, f2 = 1;
    int f;
    f = b<3  ?  1  :  f1+f2;
    f2 = f1;
    f1 = f;
    return f;
}
```

جرى الإعلان عن المتغيرين f2, f1 من النوع الصحيح الساكن مع إسناد القيمة 1 لكل منهما واستخدام مؤثر الشرط (?) لمقارنة قيم معامل الدالة المتغير b فإذا كان الشرط b<3 فيتم تخصيص القيمة 1 للمتغير f ، أما إذا كان غير ذلك فيتم تخصيص قيمة حاصل جمع آخر قيمتين للمتغير f وإسناد قيمة المتغير f1 للمتغير f2 وقبل الرجوع إلى مكان الاستدعاء بقيمة المتغير f يتم إسناد هذه القيمة للمتغير f1 ، وأخيرا فإذا أعطينا القيمة 10 إلى المتغير n فسوف يكون ناتج البرنامج كالتالي :-

The Fibonacci ==> 1+1+2+3+5+8+13+21+34+55+89

8.9) المؤشرات *Pointers*

المؤشر هو ذلك المتغير الذي يشير إلى موقع متغير بالذاكرة وليس لاسم متغير، وحتى نتحصل على المؤشر يجب استخدام :-

1) الرمز (&) ووضعه قبل المتغير لتوضيح عنوان المتغير في الذاكرة حيث تخزن المتغير .
2) الرمز *) هو المتمم لوظيفة الرمز (&) حيث ترجع قيمة المتغير .

مثال (9-8-1)

البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    int *a;
    int b = 55;
    a = &b;
    printf("\nThe value of *a ==>%d", *a);
}
```

فيه تم إشهار المتغير a وهو مؤشر يؤشر إلى قيمة من النوع الصحيح int.

جملة التخصيص :

```
a=&b;
```

تعني تخصيص موقع أو عنوان المتغير b إلى المتغير a أو بمعني آخر يشير المؤشر a ليدل على موقع الذاكرة المخصص للمتغير b .

وعليه ستكون نتيجة تنفيذ هذا البرنامج السطر الآتي :-

```
The value of *a ==>55
```

مثال (9-8-2)

المطلوب متابعة البرنامج الآتي ومقارنة نتائج تنفيذه .

```
#include <stdio.h>
main()
{

  int a = 55;
  int *pa = &a;
  printf("\nThe value of a ==>%d", a);
  printf("\nThe value of *pa ==>%d", *pa);
  *pa = 100;
  printf("\nThe value of a ==>%d", a);
  printf("\nThe value of *pa ==>%d", *pa);

}
```

ناتج تنفيذ هذا البرنامج يولد الآتي :-

```
The value of a ==>55
The value of *pa ==>55
The value of a ==>100
The value of *pa ==>100
```

9.9) الدوال والمؤشرات Functions and Pointers

في البرامج السابقة وخصوصا عند استخدام الدوال لم يكن في مقدورنا الرجوع إلا بقيمة واحدة فقط ، وعليه فقد يفرض علينا نقل بيانات بين الدالة وبين النقطة التي حدثت فيها الإشارة إلى هذه الدالة وبالعكس .

وحتي تكون الدالة قادرة على الحصول على البيانات من المكان الذي تم فيه استدعاؤها وبالتالي إعادة البيانات إلى تلك النقطة ، فيجب استخدام المؤشرات (Pointers) لفعل هذه الأشياء.

مثال (9-9-1)

دعنا الآن نقوم بكتابة دالة مهمتها استقبال متغيرين من النوع الحرفي (String) مع استبدال قيمة هذين المتغيرين والرجوع بهما إلى نقطة الاستدعاء وطباعتهما .

```
#include <stdio.h>
change_xy(char *a, char *b)
{
    char *c;
    *c = *a;
    *a = *b;
    *b = *c;
    return 0;
}
main()
{
    char *x,*y;
    x = "PLEASE";     y = "WAIT";
    printf("FUNCTION\t   X\t\t Y\n");
    printf("------------------------\n");
    printf("BEFOR CALL\t%s\t\t%s\n", x, y);
    printf("------------------------\n");
    change_xy(x, y);
    printf("AFTER CALL\t%s\t\t%s\n", x, y);
    printf("------------------------\n");

}
```

وقت التنفيذ يكون الناتج هو :

FUNCTION	X	Y
BEFOR CALL	PLEASE	WAIT
AFTER CALL	WLEASE	PAIT

الشرح

وضعت الدالة الفرعية change_xy قبل الدالة الرئيسية وهذا جائز وبالرغم من أن المعاملات جميعها من نفس النوع في الدالة والنقطة التي حدث عندها الإشارة إلى الدالة ، حيث مررت قيمة المتغير x إلى المتغير a وقيمة المتغير y إلى المتغير b وجرى تبديل المتغيرين b, a في الدالة عن طريق متغير ثالث c لكن هذا التبديل لم يحدث الا في الحرف الأول فقط .

والسبب في ذلك هو أننا تعاملنا مع قيم هذه المتغيرات وليس مع عناوينها ولمعالجة هذا يجب :-

1) استعمال المؤشرات بوضع الرمز (&) أمام كل من المتغيرين x, y في نقطة الاستدعاء لتحويلها إلى مؤشرات .
2) يجب وضع الرمز (*) أمام المتغيرات c, b, a بداخل الدالة لتصبح مؤشرات تشير إلى عناوين هذه المتغيرات .

عموما فإنه باستعمال المؤشرات أي تغيير في عناصر برنامج الدالة الفرعية المستدعاة يقابله نفس التغيير في عناصر الدالة الرئيسية عند رجوعها.

مثال (9-9-2)

يمكن إعادة كتابة البرنامج بالمثال السابق للحصول على النتائج الصحيحة.

```
#include <stdio.h>
change_xy(char *a, char *b)
{
    char *c;
    *c = *a;
    *a = *b;
    *b = *c;
    return 0;
}
main()
{
    char *x,*y;
    x = "PLEASE";     y = "WAIT";
    printf("FUNCTION\t   X\t\t Y\n");
    printf("-----------------------\n");
    printf("BEFOR CALL\t%s\t\t%s\n", x, y);
    printf("-----------------------\n");
    change_xy(&x, &y);
    printf("AFTER CALL\t%s\t\t%s\n", x, y);
    printf("-----------------------\n");
}
```

لقد تمت عملية تبديل المتغيرين عن طريق الدالة وذلك باستخدام المؤشرات ، وهذه هي النتائج .

```
FUNCTION      X          Y
---------------------------------
BEFOR        PLEASE     WAIT
---------------------------------
AFTER        WAIT       PLEASE
---------------------------------
```

مثال (9-9-3)

البرنامج الآتي يقوم بعملية ترتيب أي ثلاث قيم صحيحة تصاعديا بإستخدام الدالة .

```
#include <stdio.h>
   /* arrange three data in ascending order */
main()
{
   int a, b, c;
   void change(int *, int *);
   printf("   Arrange three data items\n");
   printf("   ************************\n");
   printf("\nType three data items  : ");
   scanf("%d %d %d", &a, &b, &c);
   if( a>b )
     change(&a, &b);
   if( b>c )
   {
     change(&b, &c);
     if( a>b ) change(&a, &b);
   }
   printf("Data in ascending order : ");
   printf("%d %d %d\n", a, b, c);
}

void change(int *i, int *j)
{
   int temp;
   temp = *i;
   *i = *j;
   *j = temp;
   return ;
}
```

إذا نفذ هذا البرنامج وأدخلت القيم المناسبة فسيكون الناتج مشابها للآتي:-

```
Arrange three data items

************************

Type three data items : 99 55 22
Data in ascending order : 22 55 99
```

في هذا البرنامج تم استدعاء الدالة :

```
change(&a, &b);
```

لأكثر من مرة حيث تم استخدام المؤشرات عند استدعائها وذلك بوضع الرمز (&) أمام المتغيرات المستعملة b, a حتي يتم تحويلها إلى مؤشرات .

أما في الدالة الفرعية وهي :

```
void change(int *i, int *j)
```

فقد تم الإعلان عن مؤشرين من النوع الصحيح j, i وهما مؤشران يشيران إلى عنواني المتغيرين j, i وبالتالي ما يحدث للمتغير i يحدث للمتغير المقابل له a ونفس الشيء يطبق على المتغير j والمتغير b .

مثال (9-9-4)

البرنامج الآتي يوضح العلاقة بين الدالة الرئيسية والدالة الفرعية باستخدام المؤشرات .

```
#include <stdio.h>
main()
{
  int a = 55;
  int pointer(int *a); /*  Prototype the function  */
  printf("\nThe value of a befor call ==>%d", a);
  pointer(&a);
  printf("\nThe value of a after call ==>%d", a);
}
int pointer(int *ptra)
{
  printf("\nThe value of pointer a at function ==>%d", *ptra);
   *ptra = 99;
}
```

بهذا البرنامج أعْلِن عن المتغير a من النوع الصحيح ثم طبعت قيمته وهي 55 ثم أُرسلت قيمة هذا المتغير إلى الدالة pointer عن طريق المؤشر ptra حيث طبعت قيمته هناك ثم أسندت القيمة 99 إلى ptra الذي هو مؤشر يؤشر

إلى قيمة صحيحة وبالتالي عند رجوع الدالة إلى موقع الاستدعاء طبعت هذه القيمة أي أن الناتج سيكون كالآتي :-

```
The value of a befor call ==>55
The value of pointer a at function ==>55
The value of a after call ==>99
```

مثال (9-9-5)

المطلوب كتابة برنامج كامل لإدخال الدرجة التي تحصل عليها الطالب في الامتحانين الأول والثاني بفصل دراسي به 5 طلبة مع :-

1) استدعاء دالة فرعية مهمتها حساب أكبر درجة امتحان ومتوسط الدرجتين مع حالة الطالب على النحو الآتي :-

* ناجح (pass) إذا كان المتوسط أكبر من أو يساوي 50 .
* راسب (fail) إذا كان المتوسط أقل من 50 .

2) استدعاء دالة فرعية مهمتها طباعة كل البيانات السابقة التي تخص الطالب .

```
#include <stdio.h>
#include <conio.h>
     /* max,avg,status are external variables */
float max, avg;
char *status;
main()
{
    int i, no;
    float t1, t2;
    float calculate(float *t1, float *t2);
    float print_them(float *t1, float *t2);
    clrscr();
    printf("Give number of students ");
    printf("and their tests:\n");
    scanf("%d", &no);
    printf("\n NUMBER  TEST1  TEST2");
    printf("  MAX   AVG  STATUS");
```

```
    printf("\n----------------------");
    printf("----------------------\n");
    for(i = 1 ; i <= no ; i++)
    {
      scanf("%f %f", &t1, &t2);
      calculate(&t1, &t2);
      print_them(&t1, &t2);
    }
    getch();
    return 0;
}
```

في هذا البرنامج تم الإعلان عن المتغيرات الخارجية max, avg من النـوع الحقيقي و status من النـوع الحرفـي وذلـك قبـل بدايـة الدالـة الرئيسـية ()main وعليه تكون جميع هذه المتغيرات شاملة ومعروفة في الدالة الرئيسـية وكذلـك بقية الدوال المستخدمة من قبل الدالة الرئيسـية ، وبعـد قـراءة بيانـات الطـالب الأول تم استدعاء :

أولاً: الدالة calculate التي هى على النحو الآتي :-

```
float calculate(float *m1, float *m2)
{
    float sum;
    max = m1 > m2 ? *m1 : *m2;
    sum = *m1 + *m2;
    avg = sum/2;
    status = avg >= 50 ?  "PASS"  :  "FAIL";
    return 0;
}
```

حيث مررت لهـا قيـم الامتحـانين t1، t2 مـن الدالـة الرئيسية عـن طريـق المؤشرات (pointers) ومن ثم جرى إيجاد :-

1) أكبر امتحان وتخصيصه للمتغير الخارجي max .

2) حالـة كـون الطـالب ناجحـاً pass أو راسـباً fail وتخصيصـه للمتغـير الخارجي status وذلك بناء على متوسط الامتحانين .

ثانياً: الدالة print_them التي هي كالآتي :-

```
float print_them(float *n1, float *n2)
{
    static int counter = 1;
    printf("\n  %d\t %.1f\t", counter, *n1);
    printf("  %.1f\t %.1f\t ", *n2, max);
    printf("%.1f\t %s\n", avg, status);
    counter++;
    return 0;
}
```

حيث تم تمرير البيانات عن طريق المؤشرات مع الإعلان عن المتغير الساكن counter كعداد حيث أعطي القيمة المبدئية 1 ثم طبعت المعلومات التي تخص الطالب رقم 1 مع الامتحان الأول والثاني عن طريق المتغيرين n2, n1 وأكبر درجة والمتوسط وحالة الطالب عن طريق المتغيرات الخارجية status, avg, max ثم ازدادت قيمة المتغير counter بمقدار 1 فأصبحت 2 وعند استدعاء هذه الدالة للمرة الثانية تبقي قيمته 2 وعليه سوف تطبع المعلومات التي تخص الطالب رقم 2 ، وهكذا تستمر عملية الاستدعاء حتي نهاية البرنامج .

وإذا ما تم تنفيذ الدالة الرئيسية مع الدوال الفرعية وأُدخلت البيانات المناسبة بعد الرسالة الآتية :-

```
Give number of students and their tests:
5  72  71    45  48    90  86    50  51    25.5  22.5
```

فسوف يكون رد الحاسب طباعة النتائج المشابهة للآتي :-

NUMBER	TEST1	TEST2	MAX	AVG	STATUS
1	72.0	71.0	72.0	71.5	pass
2	45.0	48.0	48.0	46.5	fail
3	90.0	86.0	90.0	88.0	pass
4	50.0	51.0	51.0	50.5	pass
5	25.5	22.5	25.5	24.0	fail

10.9) تمرينات Exercises

1) اكتب برنامجا يقرأ سلسلة حرفية ثم يستدعي دالة مهمتها الرجوع بكلمة yes اذا كان الرمز المدخل موجوداً في السلسلة وبكلمة no اذا كان غير ذلك، ثم يستدعي دالة أخرى لإيجاد عدد تكرار هذا الحرف بالسلسلة .

2) صف مخرجات البرامج الآتية :-

a)

```
char f_2(char * c1, char *c2)
{   *c1+=2;  *c2-=1;
   return *c1== *c2? *c1 : *c2;
}
int f_1(char m, char n)
{    m += 2;  n -= 1;
    if(m == n)
          return m += 1;
    else
          return n += 2;
}
main()
{
  char x = 'A', y = 'D', a='X',b='B',c;
  x = f_1(x,y); printf("X=%c", x);
  c=f_2(&a,&b); printf("X=%cY=%cC=%c",a,b,c);
}
```

b)

```
fun(int *a, int *b)
{
      int c;
      c=a;  a=b;  b=c;
 }
main()
{
      int x,y; x=10; y=11;
      fun(&x, &y);
      printf("\nX=%d  Y=%d", x, y);
}
```

c)

```
int x = 3;
main()
{
      int i;
      int fun(int i);
      for(i = 1 ; i <= 5 ; i++)
         printf("\nX=%d F=%d", x, fun(i) );
}

int fun(int b)
{
    static int c = 1;
    c += x+b;
    x -= b;
    return c;
}
```

d)

```
int i;
main()
{
      int fun(int );
      for(i = 7; i >= 2; i-=2)
         printf("FUN = %d\n", fun(i) );
}
int fun(int b)
{
    static int a = 5;
    a += b-1;
    return (a++ % 2 == 0 ? 1 : 0);
}
```

e)

```
void cat(int n);
void rat(int *n);
int *a;
main()
{
    int b = 2, c = 3;
    a = &c;
    printf("\nA=%d ",*a);
    cat(*a);
    printf("A=%d\n",*a);
    {
      int a = 4, c = 5;
      rat(&b);
      printf("A=%d B=%d C=%d", a, b, c);
      {
        int a = 6, b = 7, c = 8;
        cat(c);
        printf("\nA=%d B=%d C=%d", a, b, c);
      }
    }
    rat(a);
    printf("\nA=%d B=%d C=%d", *a, b, c);
}
void cat(int n)
{
    *a != 0 ? --(*a) : ++(*a);
    rat(&n);
}
void rat(int *n)
{
    if( *n)
      *n = *n+ *a;
    else
      n = a;
}
```

3) عن طريق الدالة ، أعد كتابة تمرين (6) بالفصل الخامس .

4) اكتب برنامجا يستدعي دالة لحساب :

$$sum=\frac{2}{3}+\frac{3}{5}+\frac{4}{7}+.....+\frac{n+1}{2n+1}$$

5) أعد كتابة تمرين (15) بالفصل الخامس وذلك باستخدام المتغيرات الخارجية والمتغيرات الساكنة .

6) أعد كتابة البرنامج في التمرين (1) باستخدام الدالة الفارغة على أن يتم استدعاؤها ثلاث مرات من قبل البرنامج الرئيسي .

7) اكتب دالة تحت اسم find تستقبل متغيرين S2, S1 حيث ترجع بموقع أول حرف من S1 في S2 أو ترجع بالقيمة 0 إذا كان غير ذلك .

8) اكتب برنامجا يقوم بقراءة متغيرين من نوع السلسلة الحرفية ثم يستدعي دالة تقوم بنسخ كل محتويات السلسلة الأولى بالسلسلة الثانية فيما عدا الحروف R, F, A .

9) المطلوب قراءة سلسلة حرفية لا يزيد طولها عن 25 حرفا ثم استدعاء دالتين :

* الأولى مهمتها تحديد طول هذه السلسلة .
* الثانية مهمتها طباعة السلسلة عكسيا .

10) اكتب برنامجا لادخال حرف عن طريق لوحة المفاتيح وباستخدام الدوال اعمل الآتي :-

* تحويل الحرف الكبير إلى حرف صغير .
* تحويل الحرف الصغير إلى حرف كبير .
* ترجيع رقم (0) في حالة كون الحرف المدخل رمزا خاصا والرقم (1) في حالة كونه من النوع الرقمي .

11) اكتب برنامجا لقراءة عدد صحيح num ثم كتابة دالة مهمتها إيجاد وطباعة الأرقام الأولية أي العدد الذي يقبل القسمة على الواحد وعلى نفسه فقط وذلك من 1 الى num .



13) اكتب برنامجا يقرأ قيمتين الاولي من النوع الصحيح الموجب n والثانية من النوع الحقيقي x ويستدعي دالة power التي مهمتها الرجوع بقيمة:-

$$\frac{1}{x^n}$$

14) صمم برنامجاً لقراءة اطوال اضلاع مثلث c,b,a تم احسب مساحته area عن طريق المعادلة التالية :-

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

حيث

$$s = \frac{a+b+c}{2}$$

على ان تحسب قيمة s باستخدام دالة ss والمساحة باستخدام دالة aa .

15) صمم برنامجاً كاملا وظيفته قراءة درجات لفصل به N من الطلبة استدعاء دالة تقوم بحساب متوسط درجات الطلبة وإيجاد أكبر درجة على ان تطبع المطاليب بالدالة الرئيسية باستعمال المتغيرات الخارجية أولاً وعن طريق المؤشرات ثانياً.

16) تتبع البرنامج الآتي:

```
int  a=4;
Fun_2()
{
     int x = 7, y = 11;    int *ptr1, *ptr2 ;  ptr1 = &x;      ptr2 = &y;
     *ptr1 = *ptr2+2;     *ptr2 = *ptr1+2;
     printf("X=%d  B=%d", x, y);
}
int Fun_1(int x, int *p, char *y)
{     a=*p;    *p +=x;   a=*p + 3;  ++*y;  }
main ()
{     int b=5; char c='B';
      Fun_1(a, &b, &c);
      printf("A=%d  B=%d  C=%c",a, b,c);
      Fun_2();
}
```

10

الفصل العاشر

دالة الإعادة والماكرو والدوال الرياضية

1.10) دالة الإعادة الذاتية Recursion Function

وهي الدالة الفرعية التي يمكن أن تستدعي نفسها ، ويجب عند استخدام هذا النوع من الدوال وجود شرط معين ينهي عملية الاستدعاء بصورة دائمة وإذا لم يتحقق الشرط يتم استدعاء الدالة لنفسها في النقطة المناسبة .

مثال (1-1-10)

البرنامج التالي يقوم باستدعاء دالـة الإعـادة التـي مهمتها حسـاب مجمـوع أول 5 أعداد صحيحة .

```
#include <stdio.h>
main()
{
    int sum, n = 5;
    int sum_it(int);
    printf("The following is the sum of \n");
    printf("the first 5 positive integers\n");
    sum = sum_it(n);
    printf(" WHICH EQUAL TO %d", sum);
    return 0;
}

int sum_it(int m)
{
    if( m == 1 )
    {
      printf("%d", m);
      return m;
    }
    else
```

```
        printf("%d+", m);
      return ( m + sum_it (m-1) );
  }
```

هنا تُمَرر قيمة المتغير n إلى معامل الدالة sum_it المتغير m ، فإذا كان شرط جملة if صحيحا عندها ترجع الدالة بنفس القيمة المستقبلة من الدالة الرئيسية main() أما إذا كان الشرط خاطئا فيتم تنفيذ جملة الطباعة وتنفيذ الجملة:

```
return ( m + sum_it (m-1) );
```

وهي تعني أضف قيمة m إلى الدالة مع استدعائها لنفسها مرة أخرى بعد طرح قيمة 1 من المتغير m وإرسال الناتج إلى دليل الدالة m مرة أخرى ويستمر هذا حتي تصبح قيمة المتغير m تساوي 1 عندها يتحقق شرط if حيث تضاف القيمة النهائية للمتغير m وبالتالي يتم الرجوع بها إلى نقطة الاستدعاء وتطبع هناك .

تنفيذ هذا البرنامج سيعطي الآتي :-

```
The following is the sum of
the first 5 positive integers
5+4+3+2+1 which equal to 15
```

مثال(10-1-2)

المطلوب كتابة برنامج مهمته استقبال سلسلة حرفية ثم استدعاء دالة الإعادة الذاتية لتحديد طول هذه السلسلة وطباعتها عن طريق دالة إعادة ذاتية أخرى .

الحل

يمكن كتابة دالة الإعادة الأولي التي قد تكون كالآتي :-

```
int str_length(char *str)
{
    if( *str )
      return( 1 + str_length (str+1) );
    else
      return 0;
}
```

تبدأ هذه الدالة باستقبال السلسلة الحرفية وتخزينها في المؤشر str فإذا كانت السلسة فارغة عندها ترجع الدالة بقيمة 0 الناتجة من الجملة

```
return 0;
```

أما إذا كانت غير ذلك عندها تنفذ الجملة

```
return( 1 + str_length (str+1) );
```

وتعني أنه في كل مرة تستدعي فيها الدالة str_length نفسها يضاف إليها القيمة 1 وفي نفس الوقت تزداد السلسلة بمقدار 1 وهو يعني الانتقال إلى الحرف التالي بالسلسلة ، ويتكرر هذا حتي يصل المؤشر إلى رمز نهاية السلسلة (\0) وبالتالي ترجع الدالة بقيمة واحدة من النوع الصحيح الذي يمثل طول السلسلة .

أما فيما يخص دالة الإعادة الثانية فهي :

```
int print_str(char *s)
{
    if( *s )
    {
       printf("%c", *s);
       return ( print_str ( ++s ) );
    }
    return 0;
}
```

هذه الدالة تحتوي على معامل واحد s حيث تمرر عبره السلسلة الحرفية ويجري طباعة الحرف الأول ثم تُستَدعى نفس الدالة بعد الانتقال إلى الحرف الآتي بالسلسلة وهكذا حتي الوصول إلى رمز النهاية (\0) .

وفيما يلي الدالة الرئيسية مع الدوال .

```
#include <stdio.h>
main()
{
    int j;
    char *k;
    int str_length(char *);
    int print_str(char *);
    printf("\nType your string : ");
    gets(k);
    j = str_length(k);
    printf("\nThe string you typed is : ");
    print_str(k);
    printf("\nIt has %d characters.", j);
    return 0;
}

int str_length(char *str)
{
    if( *str )
      return( 1 + str_length (str+1) );
    else
      return 0;
}

int print_str(char *s)
{
    if( *s )
    {
       printf("%c", *s);
       return ( print_str ( ++s ) );
    }
    return 0;
}
```

وفيما يلي إدخال البيانات المناسبة والمخرجات وقت التنفيذ .

Type your string : WELCOME TO COMPUTER CENTER.

The string you typed is : WELCOME TO COMPUTER CENTER.
It has 27 characters.

مثال (3-1-10)

المطلوب كتابة برنامج كامل مهمته الآتي :-

1) استدعاء الدالة الذاتية least_common_multiple ووظيفتها الرجوع بقيمة المضاعف المشترك الأصغر (LCM) لعددين من النوع الصحيح int فمثلا :

إذا كان لدينا عددان صحيحان 10 ,15 فإن المضاعف المشترك الأصغر LCM للعددين هو :

$$LCM = \frac{15 \times 10}{GCD\,(15,10)} = \frac{15 \times 10}{5} = 30$$

وهذه الدالة يمكن كتابتها كالآتي :-

```
int least_common_multiple(int a,int b)
{
    int c, d;
    c = a;
    d = b;
    return( c * d / greatst_common_divisor (a, b) );
}
```

حيث يمرر لهذه الدالة عددان صحيحان يتم وضعهما في المتغيرين الصحيحين b, a وحتي يمكن استخدام هذين المتغيرين لاحقا ، تم وضع قيمة المتغير a في المتغير الداخلي c وقيمة المتغير b في المتغير d ثم تأتي الجملة:

```
return( c * d / greatst_common_divisor (a, b) );
```

وفيها تحدث عملية ضرب المتغيرين d, c أولا وقبل إجراء عملية القسمة تستدعي الدالة greatst_common_divisor التي بدورها تقوم بإرجاع القاسم المشترك الأكبر (GCD) للمتغيرين b, a .

وقبل كتابة هذه الدالة ، يمكن إعطاء فكرة عن كيفية حساب القاسم المشترك الاكبر كالآتي :-

1) نقوم بتحليل كل عدد إلى قواسمه على حدة ثم نبحث عن القواسم التي يشترك فيها العددان .
2) نضرب القواسم المشتركة في بعضها فينتج عنها القاسم المشترك الأكبر .

فمثلا : العدد 24 ناتج من ضرب قواسمه 3x2x2x2

والعدد 28 ناتج من ضرب قواسمه 7x2x2

ومن هنا فإن العددين مشتركان في القاسم (2) مرتين .

إذا 2X2 = 4 وهو القاسم المشترك الأكبر للعددين 24, 28 .

والآن ننتقل إلي كتابة دالة الإعادة الذاتية التي تقوم بهذه المهمة .

```
int greatst_common_divisor(int x, int y)
{
    int temp;
    if( x == y )
      return x;
    else
      if( y>x )
      {
        temp = y;
        y = x;
        x = temp;
      }
    return( m * n / greatst_common_divisor(x-y , y) );
}
```

وفيها إذا كان العددان المرسلان من الدالة least_common_multiple متساويين فيتم الرجوع بقيمة المتغير x إلى نقطة استدعائها ، أما إذا كانا غير ذلك فعندها تقارن القيمتان مع بعضهما وفي حالة أن القيمة الثانية أكبر من القيمة الأولى تتم عملية التبادل باستحداث متغير مؤقت temp ثم تستدعي هذه الدالة نفسها مرة أخرى عن طريق الجملة

```
return( m * n / greatst_common_divisor(x-y , y) );
```

وتستمر عملية الاستدعاء مع طرح قيمة y من قيمة x حتي يتساويان وعندها تتوقف الدالة عن تكرار نفسها وترجع بقيمتها إلى مكان استدعائها أي عند الجملة

```
return(c*d/greatst_common_divisor(x, y));
```

والموجودة في الدالة السابقة

```
lease_common_multiple(a, b)
```

أخيرا وحتي يكون العمل متكاملا من جميع جوانبه يتحتم كتابة الدالة الرئيسية التي تقوم بقراءة عددين من النوع الصحيح ثم استخدام الدوال السابقة للحصول على الناتج النهائي ، وفيما يلي البرنامج كاملا .

```
#include <stdio.h>
const int SIZE = 3;
int m, n;
int greatst_common_divisor(int, int);
main()
{
    int i, L, G;
    int least_common_multiple(int, int);
    int out_put(int, int, int, int);
    printf("\nEnter two positive integer ");
    printf("numbers(%d times):\n", SIZE);
    for(i = 1 ; i <= SIZE ; i++)
    {
       scanf("%d%d", &m, &n);
       L = least_common_multiple(m, n);
```

```
        G = greatst_common_divisor(m, n);
        out_put(m, n, L, G);
    }
    return 0;
}

int out_put(int p, int q, int s, int t)
{
    printf("\n#-------------------------#");
    printf("\n#\tLCM (%d,%d) = %d  #", p, q, s);
    printf("\n#\tGCD (%d,%d) = %d  #", p, q, t);
    printf("\n#-------------------------#");
    return 0;
}

int least_common_multiple(int a, int b)
{
    int c, d;
    c = a;
    d = b;
    return( c*d / greatst_common_divisor(a, b) );
}

int greatst_common_divisor(int x, int y)
{
    int temp;
    if( x == y )
      return x;
    else
      if( y>x )
      {
        temp = y;
        y = x;
        x = temp;
      }
    return( m * n / greatst_common_divisor(x-y, y) );
}
```

عند تنفيذ هذا البرنامج سوف نحصل على النتائج الآتية :-

```
Enter tow positive integer numbers(3 times):
72  24  15  30  91  169
```

```
#-------------------------------- #
#      LCM (72,24) = 72      #
#      GCD (72,24) = 24     #
#-------------------------------- #
#-------------------------------- #
#      LCM(15,30) = 15      #
#      GCD(15,30) = 30      #
#-------------------------------- #
#-------------------------------- #
#      LCM(91,169) = 13     #
#      GCD(91,169) = 1183 #
#-------------------------------- #
```

2.10) المعالج الأولي *Preprocessor*

في لغة C يوجد الأمر أو الموجه #define الذي يعتبر من أوامر المعالج الأولي وهو يقوم بإنشاء معرفات تسمي للثابت الرمزي (Symbolic constant) حيث يمكن وضعه في أي مكان من البرنامج على أن يكون ذلك قبل استعماله ولكن يستحسن وضعه قبل بداية الدالة الرئيسية main() .

مثال (10-2-1)

الأمر :

```
#define  MAX_LENGTH 50
```

يسبب استبدال الثابت الرمزي MAX_LENGTH بالقيمة 50 كلما وجد في البرنامج .

والثابت الرمزي :-

1) يمكن كتابته بالحروف الصغيرة أو الكبيرة

2) لا يمكن تغيير قيمته سواء بالزيادة أو النقصان أثناء تنفيذ البرنامج .

فمثلاً من الخطأ أن نكتب الجملة مع الأمر السابق .

```
MAX_LENGTH = MAX_LENGTH + 40;
```

3) قيمة هذا الثابت الرمزي يمكن أن تكون قيمة عددية أو حرفية أو سلسله حرفية .

مثال (10-2-2)

الأوامر التالية تعتبر مقبولة :

```
#define PI    3.14159
#define  STRING  "THIS IS A STRING"
#define  SLASH   '\'
```

3.10) الماكرو *Macro*

وهو الذي يمكن استخدامه مع الأمر #define في كثير من الأحيان عوضا عن الدوال البسيطة .

مثال (10-3-1)

فيما يلي بعض الأمثلة على استخدامات الماكرو .

a)

```
#define  pay(hrs, rate)          hrs*rate+bonus
main()
{
      . . .
      bouns = 45;
      cal = pay(40, 5);
      . . .
}
```

b)

```
#define   fahr(c)              9 / 5.0 * c + 32
main()
{
    . . .
    temp = fahr(50);
    . . .
}
```

مثال (2-3-10)

لإيجاد أكبر قيمة من قيم المتغيرات z, y, x فإن البرنامج التالي يقوم بعمل ذلك .

```
#include <stdio.h>
#define print          printf("\nEnter 3 values now :");
#define large(a, b)    a>b  ? a : b.
main()
{
  int x, y, z, max;
  print;
  scanf("%d %d %d", &x, &y, &z);
  max = large(x, y);
  max = large(max, z);
  printf("Max ==>%d", max);
}
```

التنفيذ ينتج عنه الآتي :-

```
Enter 3 values now : 30 35 20
max ==>35
```

بالبرنامج وبعد الإعلان عن التوجيه #define بالصورة :

```
#define print          printf("\nEnter 3 values now :");
```

تمت عملية الاستدعاء عن طريق الجملة :

```
print;
```

ونتج عنها طباعة السطر

```
Enter 3 values now :
```

وإذا ما أدخلت القيم 30 ,35 ,20 فعندها يتم استدعاء الماكرو :

```
#define large(a, b)  a>b ? a : b
```

حيث أرسلت قيمة المتغيرين y, x إلى المتغيرين b, a وبالتالي الرجوع بأكبر قيمة باستخدام مؤثر الشرط (?) وحفظها بالمتغير max وتتكرر عملية الاستدعاء ولكن بقيمتي z, max وأخيرا طباعة الناتج وإيقاف البرنامج .

مثال (10-3-3)

المطلوب كتابة برنامج لإيجاد تربيع وتكعيب أي قيمة صحيحة يتم إدخالها.

```
#include <stdio.h>
#define squre(x)        x*x
#define cube(x)         x*squre(x)
main()
{
      int a;
      printf("Enter integer value ==>");
      scanf("%d", &a);
      printf("\nThe squre of %d ==> %d", a, squre(a) );
     printf("\nThe cube of %d  ==> %d", a, cube(a) );
}
```

الناتج وقت التنفيذ هو الآتي :-

```
The squre of 3 ==> 9
The cube of 3  ==> 27
```

مثال (10-3-4)

المطلوب إعادة كتابة البرنامج بالمثال (9-9-3) بالفصل (9) الذي مهمته قراءة ثلاث قيم صحيحة وبالتالي ترتيبها تصاعديا باستخدام الماكرو .

```
#include <stdio.h>
        /*  Program to arrange three data   */
        /*   items in ascending order      */
        /*    using define statement       */
#define  STR  "Data in ascending order :"
#define  N  6
#define swap(x, y)   z = x;  x = y;  y = z;
main()
{
    int i, a, b, c, z;
    printf("\n Arrange three data item\n");
    printf("   ***********************\n");
    for(i = 1 ; i <= N ; i++)
    {
      printf("\n Type three data items : ");
      scanf("%d %d %d", &a, &b, &c);
      if( a>b )
      {
        swap(a, b);
      }
      if( b>c )
      {
        swap(b, c);
      }
      if( a>b )
      {
        swap(a, b);
      }
      printf("%s %d %d %d\n", STR, a, b, c);
    }
}
```

وهذا هو ناتج البرنامج :

```
Arrange three data items
***********************
Type three data items : 99 55 22
Data in ascending order : 22 55 99
```

مثال(10-3-5) .

البرنامج الآتي يوضح احتواء الماكرو لأكثر من جملة واحدة .

```
/*  average of values using define  */
#include <stdio.h>
#define print  printf("\nEnter values and 0 stop ==>");
#define do_sum_count          {  sum += x;          \
                                   c += 1;          \
                                 scanf("%f", &x); \
                              }
main()
{
    int i, c = 0;
    float x, avg, sum = 0.0;
    #define write printf("The average of %d values is %.2f",c, avg);
    print;
    scanf("%f", &x);
    while( x != 0 )
        do_sum_count;
    avg = sum/c;
    write;
    return 0;
}
```

تنفيذ هذا البرنامج سيعطي النتائج المشابهة للآتي :-

```
Enter values and 0 stop ==> 4  8  6  3  5 0
The average of 5 values is 5.20
```

عن طريق جملة while مادامت قيمة المتغير المدخل x لا تساوي 0 فسيتم استدعاء الماكرو تحت اسم do_sum_count الذي يحتوي على عدد من السطور كل سطر يجب أن ينتهي بالشرطة المائلة (\) فيما عدا الأخير حيث عن طريقها يحسب مجموع القيم المدخلة وعددها مع استقبال القيمة الموالية وهكذا حتي يتم إدخال القيمة 0 عندها يحسب المتوسط avg ويُستَدْعى الماكرو write لطباعة هذا المتوسط .

4.10) الدوال العامة General Functions

تناولنا سابقا كيفية تقسيم البرنامج إلى عدد من الدوال الفرعية وبالتالي استدعائها وقت الحاجة إليها عن طريق الدالة الرئيسية ، بنفس الطريقة يمكن

كتابة كل دالة فرعية وحفظها في ملف منفصل تحت ملفات العناوين (Heades Files) يحتوي على الامتداد (h) وعليه تصبح دوال عامة يمكن استدعاؤها بواسطة التوجيه (#include) على غرار الملف stdio.h الذي يضم عددا من الدوال مثل scanf, printf وغيرها .

مثال (1-4-10)

المطلوب كتابة برنامج مهمته استقبال قيمتين من النوع الصحيح مع طباعة القيمة العظمي .

الحل

يمكن إنشاء عدد من الملفات وهي :-

1) الملف الأول تحت الاسم find_max.h ويضم الدالة find_max التي هي مبينة بالشكل التالي :-

```
/*  find_max.h  */
int find_max(int a, int b)
{
  int m;
  m = a>b ? a : b;
  return m;
}
```

ومهمتها استقبال قيمة المتغيرين b, a والرجوع بالقيمة الكبرى m .

2) الملف الثاني تحت الاسم prnt_max.h ويضم الدالة التالية :-

```
/*  prnt_max.h  */
int prnt_max(int mm)
{
  printf("The max number ==> %d\n", mm);
}
```

بحيث تطبع القيمة العظمي التي مررت إليها عن طريق المعامل mm

3) الملف الثالث تحت الاسم wait.h ويحتوي الدالة :

```
/* wait.h */
void wait()
{
    printf("Press any key to continue ..");
    getch();
}
```

التي مهمتها تعليق البرنامج حتى الضغط على أي مفتاح للاستمرار .

4) الملف الرابع تحت الاسم my_prog.c (لاحظ امتداد الملف) وبه الدالة الرئيسية التالية :-

```
/* my_prog.c */
#include <stdio.h>
#include <conio.h>
#include "find_max.h"   /* for find_max function */
#include "prnt_max.h"   /* for prnt_max function */
#include "wait.h"       /* for wait function */
main()
{
    int n1, n2, max;
    clrscr();
    printf("\n Type 2 numbers ==>");
    scanf("%d %d", &n1, &n2);
    max = find_max(n1, n2);
    prnt_max(max);
    wait();
    return 0;
}
```

وفيها تم استخدام التوجيه #include لإمكانية ربط الملفات السابقة مع هذه الدالة ، تلا ذلك استقبال القيمتين n2, n1 واستدعاء الدالة find_max الموجودة في الملف المنفصل find_max.h والرجوع بالقيمة الكبرى وتخزينها بالمتغير max ثم إرسال قيمة هذا المتغير إلى الدالة prnt_max الموجودة بالملف الثاني

prnt_max.h وبالتـالي طباعـة هـذه القيمـة وأخيراً اسـتدعاء الدالـة wait التـي بالملف wait.h بقصد الانتظار حتي الضغط على أي مفتاح للاستمرار .

عموماً عند تنفيذ الدالة الرئيسية سيتم ترجمة الأوامر بالملفات عـن طريـق المترجم (Compiler) ملفا يلي الآخر وفي حالة وجود أي خطأ في أحد الملفات فسوف يشير إليه المترجم أيضاً يمكن استدعاء الملفات ذات الامتـداد (h) مـن قبل أي مستخدم، وفيما يلي النتائج عند التنفيذ .

```
Type 2 nymbers ==> 66 55
The max number ==> 66
Press any key to continue ..
```

5.10) الدوال الرياضية *Mathematical* **Functions**

تحتوى لغة C على مجموعة من الدوال الرياضيـة كـالموجودة فـي بعـض الآلات الحاسبة الصغيرة ، وحتـي يمكـن اسـتغلالها يجـب اسـتخدام الملـف <math.h> الـذي يحتـوي علـى تعريفـات للعديـد مـن هـذه الـدوال والمدونـة بالجدول التالي :-

الدالة	الغرض منها	نوع المتغير	القيمة المرجعة
abs(x)	القيمة المطلقة	int	int
acos (x)	معكوس جيب التمام	double	double
asin (x)	معكوس الجيب	double	double
atan (x)	معكوس الظل	double	double
atan2 (y,x)	معكوس الظل y/x	double	double
cos (x)	جيب التمام	double	double
cosh (x)	جيب التمام الزائد	double	double
exp (x)	القيمة الاسية	double	double
fabs (x)	القيمة المطلقة	double	double
log (x)	اللوغاريتم الطبيعية	double	double
log 10(x)	اللوغاريتم العشرية	double	double

الدالة	الغرض منها	نوع المتغير	القيمة المرجعة
pow (x,y)	x بقوة y	double	double
sin (x)	الجيب	double	double
sinh (x)	الجيب الزائد	double	double
sqrt (x)	الجذر التربيعي	double	double
tan (x)	الظل	double	double
tanh (x)	الظل التعظيمي	double	double

مع الأخذ في الاعتبار عند استخدام الدوال tan, cos, sin أن تكون بالتقدير الدائري.

مثال (10-5-1)

البرنامج الآتي مهمته توضيح عمل الدوال ، القيمة المطلقة والجذر التربيعي ودالة الرفع للقوة .

```
#include <conio.h>
#include <process.h>  /*  for exit function  */
#include <stdio.h>
#include <math.h>  /*  for abs, sqrt and pow functions  */
/*  Program using abs,sqrt and pow functions  */
main()
{
    int i, x, op;
    double a, b, y;
    for( ; ; )
    {
      printf("\n Type 1 to get abs function ");
      printf("\n      2 to get sqrt function ");
      printf("\n      3 to get pow function ");
      printf("\n      4 to exit ");
      printf("\n\nYour selection please : ");
      scanf("%d", &op);
      switch (op)
      {
       case 1 : printf("\nEnter value of x : ");
                scanf("%d", &x);
                printf("abs(%d) ==> %d", x, abs( x ) );
                break;
       case 2 : printf("\nEnter value of y : ");
```

```
                scanf("%lf", &y);
                printf("sqrt(%.0f) ==> %.0f", y, sqrt( y ) );
                break;
        case 3 : printf("\nEnter value of x and y : ");
                scanf("%lf %lf", &a, &b);
                printf("%.0f power %.0f ==> %.0f", a, b, pow(a, b) );
                break;
        case 4 : exit(0);
      }
      getch();
    }
}
```

إذا ما نفذ هذا البرنامج فسيظهر على شاشة نظيفة قائمة تضم الدوال التي يحتويها هذا البرنامج كالآتي :-

```
Type 1 to get abs function
     2 to get sqrt function
     3 to get pow function
     4 to exit
```

وإذا ما أدخل الاختيار رقم 1 وأدخلت القيمة 777- أي إيجاد القيمة المطلقة كما يلي :-

```
Your selection please : 1
Enter value of x : -777
abs(-777) ==> 777
```

وبالتالي أعطيت الدالة القيمة الصحيحة المطلقة للعدد السالب 777- وهو 777 ، مع عدم الحصول على أصغر قيمة صحيحة سالبة يتراوح مدى الأعداد التي تأخذ 16 بت بين الرقم 32768- والرقم 32767 ، فمثلا إذا ما نفذ البرنامج السابق وأدخلت القيمة 33333- عند اختيار رقم 1 سيكون الناتج مخالفا لما تتوقعه وهو المشابه للآتي :

```
abs(-33333) ==> 32203
```

كما يمكن إيجاد الجذر التربيعي من النوع الحقيقي بالدقة المضاعفة عن طريق الاختيار 2 مع إدخال القيمة 16 للمتغير y كما يلي :-

```
Type 1 to get abs function
     2 to get sqrt function
     3 to get pow function
     4 to exit

Your selection please : 2
Enter value of y : 16
sqrt(16) ==> 4
```

اما إذا أدخل الاختيار 3 والقيمتان 2, 16 أي الحصول على مربع القيمة 16 كالآتي :-

```
Type 1 to get abs function
     2 to get sqrt function
     3 to get pow function
     4 to exit

Your selection please : 3
Enter value of x and y : 16   2
16 power 2 ==> 256
```

وهكذا يستمر تنفيذ هذا البرنامج حتي يتم إدخال الرقم 4 وبالتالي الخروج من جملة for وإيقاف التنفيذ .

6.10) تمرينات Exercises

1) اكتب برنامجاً لقراءة أطوال أضلاع مستطيلات ، باستخدام الماكرو المطلوب حساب مساحة ومحيط المستطيل ، علما بأن :

المساحة = الطول × العرض

المحيط = 2 × (الطول + العرض)

2) اكتب برنامجا لاستقبال رمز مستخدما فيه الماكرو لإرجاع القيم 1 اذا كانت القيمة موجبة 2 اذا كانت القيمة سالبه 3 اذا كانت غير ذلك .

3) صمم برنامجاً باستخدام دالة الإعادة الذاتيه أوجد مجموع الأعداد الفردية من 1 الى N .

4) اكتب برنامج يقرأ قيمة من النوع الصحيح وباستخدام الماكرو اذا كانت القيمة زوجية اطبع even واطبع odd اذا كانت غير ذلك .

5) باستخدام دالة الإعادة الذاتية ، أعد كتابة تمرين (4) بالفصل التاسع .

6) المطلوب كتابة برنامجاً كاملا لإدخال سلسلة حرفية بسطر واحد وعن طريق دالة المعاودة الذاتية اطبع هذا السطر في ترتيب عكسي بدون فراغات .

7) باستخدام المعالج الأولي ، المطلوب إعادة كتابة تمرين (6) بالفصل الخامس .

8) أعد كتابة تمرين (16) بالفصل الخامس لحساب متوسط الامتحانات لكل طالب وعدد التقديرات A، F باستخدام دالة الإعادة الذاتية .

9) باستخدام دالة الإعادة الذاتية أوجد مجموع مربع الأعداد من 1 إلى n .

10) باستخدام دالة الاعادة الذاتية POW_1 اكتب برنامج لحساب x^N بحيث تكون N عدد صحيح موجب X أي عدد كسري .

11) قم بكتابة برنامج لإيجاد أكبر قيمة وأصغر قيمة لأي قيمتين باستخدام الماكرو .

12) اكتب برنامجاً لقراءة قيمة المتغير N ثم أوجد الاتي:-

a) $S = 2! + 4! + 6! + \ldots + N!$

b) $S = \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!} \ldots \pm \frac{1}{N!}$

c) $S = \frac{1}{2!} - \frac{3!}{4} + \frac{5}{6!} - \frac{7!}{8} + \ldots \pm \frac{N!}{(N+1)}$

باستخدام الماكرو مرة والدالة الرياضية pow مرة اخرى.

13) المطلوب تتبع البرنامج الآتي أولاً ثم تحويله باستخدام الدالة العادية ثانيا.

```
#include <stdio.h>
int fun(int m,int n);
main()
{
    int n = 7;
    printf("fun=%d", fun(n, n) );
}
int fun(int m, int n)
{
    if(m>0)
        return (n + fun(m-1, n));
}
```

14) أوجد ناتج تنفيذ البرامج التالية مع إعادة كتابتها باستخدام الدالة عوضا عن الموجه define .

a)

```
#include <stdio.h>
#define write          printf("Enter 2 values -->");
#define read_cd        scanf("%f  %f", &c, &d);
#define mult(n,m)      n*m
main()
{
    int a = 5, b = 6;
    float c, d;
    write;
    read_cd;
    printf("\n%d * %d = %d", a, b, mult(a, b));
    printf("\n%.2f * %.2f = %.2f", c, d, mult(c, d));
}
```

b)

```
#include <stdio.h>
#define write(a,b)  printf("\nA=%d  B=%d", a, b);
#define use_for {  for(i = 1; i <= n; i++)        \
                    {                \
                        scanf("%d", &x);       \
                        if(x % 2 == 0)         \
                          s1++;                \
                        else                   \
                          s2++;                \
                    }                          \
                }                              \
main()
{
    int i, s1, s2, x, n = 5;
    s1 = s2 = 0;
    use_for;
    write(s1, s2);
}
```

11

الفصل الحادي عشر

المصفوفات

عند التعامل مع حجم كبير من البيانات كنا نستخدم الكثير من المتغيرات لتخزين ومعالجة تلك البيانات ، لأن كل متغير تخزن فيه قيمة واحدة فقط ، وقد كان هذا سببا في إطالة وتعقيد البرنامج في أغلب الأحيان .

ولكن باستخدام المصفوفة (Array) التي هي تركيبة بيانية يستطيع المبرمج بواسطتها استعمال متغيرات قليلة وذلك بتقسيم كل متغير إلى عدد من العناصر (Elements) المتسلسلة مما يُمَكِّنهُ من تخزين أكثر من قيمة واحدة في متغير واحد بشرط أن تكون من نفس النوع ، وهناك نوعان من المصفوفات:-

1.11) المصفوفة ذات البعد الواحد One-d*imensional Array*

1) المصفوفة العددية A*rray* N*umbers*

وهي عبارة عن صف أو عمود واحد يحتوي على مجموعة من عناصر البيانات متحدة النوع والاسم .

الشكل العام

```
type   array_name [index];
```

حيث array_name يمثل اسم المصفوفة في حين index يمثل دليل هذه المصفوفة ، أي عدد عناصرها الذي يجب أن يكون رقما أو متغيرا يحتوي على رقم صحيح موجب ومحصور بين القوسين [ ] .

مثال (11-1-1)

قد يأخذ دليل المصفوفة تعبيرا حسابيا صحيحا عوضا عن الثوابت كما هو الموضح فيما يلي :-

```
printf("%d",new_row [2] );
sum += price[k+1];            أو
MAT[N] = 25                   أو
```

حيث تدل القيمة الموجودة بين القوسين [ ] على ترتيب العنصر في المصفوفة .

مثال (11-1-2)

الامر

```
int  list [15];
```

يعني تعريف مصفوفة أحادية تحت اسم list تحتوي على 15 عنصرا من النوع الصحيح int وهذه العناصر بدايتها من list [0] وآخرها list [14] حيث يشار إلى أي عنصر في المصفوفة باستخدام اسم المصفوفة أولا ودليل العنصر داخل المصفوفة بين القوسين [ ] ثانيا .

مثال (11-1-3)

البرنامج التالي

```
#include <stdio.h>
main()
{
      int i, list [6];
      for(i = 0 ; i <= 6 ; i++)
         list [i] = i;
      for(i = 0 ; i <= 6 ; i++)
         printf("I=%d  ", list [i] );
}
```

فيه تم شحن المصفوفة list بالقيم الصحيحة الموجبة من 0 إلى 5 والناتج من استخدام الدليل i المتحصل عليه من جملة for وبالتالي احتوت هذه المصفوفة على 6 قيم صحيحة وهي :

I=0 I=1 I=2 I=3 I=4 I=5 I=6

مثال (4-1-11)

ماذا يحدث إذا ما تعدى دليل المصفوفة حجمها ؟ البرنامج التالي يوضح هذا .

```
#include <stdio.h>
main()
{
      int i, list [6];
      for(i = 0 ; i <= 7 ; i++)
         list[i] = i;
      for(i = 0 ; i <= 7 ; i++)
         printf("I=%d  ", list [i]);
}
```

هنا بدأ دليل المصفوفة list الذي يمثله المتغير i من 0 حتي 7 وهي 8 عناصر في حين أن المصفوفة حجمها 6 وعليه سينتج عند تنفيذ هذا البرنامج ما يلي :-

I=0 I=1 I=2 I=3 I=4 I=5 I=6 Abnormal program termination

وكما نلاحظ في آخر هذه النتيجة هناك رسالة تقول بأن إيقاف البرنامج غير طبيعى ، وعلى هذا يجب الأخذ في الاعتبار تحديد بداية ونهاية الدليل المستخدم مع المصفوفة ، على ألا يتعدى الدليل حجم المصفوفة ، وإلا كان الناتج خاطئا كما لاحظنا .

مثال (5-1-11)

لنفرض أننا نريد قراءة رقم الطالب ودرجته التي تحصل عليها في امتحان الحاسب الآلي وذلك لفصل به 5 طلبة ، والمطلوب طباعة رقم الطالب ودرجته مع حالته بالنسبة لمتوسط الفصل .

```
#include <stdio.h>
#define SIZE 5
main()
{
    float avg, sum, grade[SIZE];
    long i, id_no[SIZE];
    sum = 0.0;
    printf("\nType number and grade of ");
    printf("%d students please :\n\n", SIZE);
    for(i = 0 ; i < SIZE ; i++)
    {
      scanf("%ld%f", &id_no[i], &grade[i]);
      sum += grade[i];
    }
    avg = sum/SIZE;
    printf("The class average is %.2f", avg);
    printf("\n--------------------------------");
    for(i = 0 ; i < SIZE ; i++)
    {
      printf("\nID#  %ld", id_no[i]);
      printf("  GRADE = %.2f", grade[i]);
      printf("  DIFFERENT = %.2f", grade[i] - avg );
    }
    printf("\n---------------------------------------");
}
```

بدأ البرنامج بالإعلان عن المصفوفة

```
float grade[SIZE];
```

وهو يعني حجز مكان في ذاكرة الحاسب كمصفوفة تحت اسم grade تحفظ فيه عدد 5 قيم من النوع الحقيقي مع إمكانية الرجوع لهذه القيم وقت الحاجة إليها .

أما الإعلان

```
long id_no[SIZE];
```

يعني أن هناك مصفوفة تحت اسم id_no من النوع الصحيح الطويل حيث تضم 5 عناصر بقصد تخزين أرقام قيد الطلبة .

بعد إصدار بعض الرسائل التوضيحية جاءت جملة for ومهمتها تكرار قراءة رقم القيد والدرجة وتخزينها في الأماكن التي يشير إليها دليل المصفوفة المعنية بداية من 0 إلى 4 مع حساب مجموع درجات كل الطلبة ، وبعد حساب المتوسط ، استخدمت جملة for لإخراج البيانات التي تم إدخالها سالفا مع الفرق بين الدرجة ومتوسط الفصل .

باستخدام المصفوفة نرى مدى إمكانية معالجة بيانات كثيرة أي إذا أردنا تنفيذ هذا البرنامج لفصل به أكثر من 5 طلبة ، فما علينا إلا أن نغير قيمة الثابت SIZE بالبرنامج فقط .

وإذا ما نفذ هذا البرنامج وتم إدخال أرقام القيد مع الدرجة لكل طالب بعد الرسالة التوضيحية :

```
Type number and grade of 5 students please :
994001 64   993223 75   993106 50   993076 49   993024 45
```

سيلي ذلك متوسط الفصل ثم جدول يبين رقم كل طالب ودرجته والفرق بين الدرجة والمتوسط كما يلي :-

```
The class average is 56.60
-----------------------------------------------------------------
ID# 994001  GRADE = 64.00  DIFFERENT = 7.40
ID# 993223  GRADE = 75.00  DIFFERENT = 18.40
ID# 993106  GRADE = 50.00  DIFFERENT = -6.60
ID# 993076  GRADE = 49.00  DIFFERENT = -7.60
ID# 993024  GRADE = 45.00  DIFFERENT = -11.60
-----------------------------------------------------------------
```

مثال (6-1-11)

الهدف من البرنامج التالي هو قراءة M من القيم الصحيحة لاتزيد عن 50، وتخزينها في مصفوفة mat ثم إيجاد القيم الموجبة فقط وحفظها في مصفوفة أخرى other وطباعتها .

```
#include <stdio.h>
#include <process.h>
#define M 50
main()
{
   int mat[M], other[M], n, i, counter = 0;
   printf("Enter size of the array");
   printf(" less than %d: ", M);
   scanf("%d", &n);
   printf("\nThe array MAT it looks like :- ");
   for(i = 0 ; i < n ; i++)
   {
     scanf("%d", &mat[i]);
     printf("\nMAT[%d] ==> %d", i, mat[i]);
   }
   for(i = 0 ; i < n ; i++)
    if( mat[i] > 0 )
    {
        other[counter] = mat[i];
        counter++;
    }
   if( counter == 0 )
      exit(0);
   printf("\n\nWhile array OTHER it looks like :- ");
   for(i = 0 ; i < counter ; i++)
     printf("\nOTHER[%d] --> %d", i, other[i]);
}
```

وقت تنفيذ البرنامج وإدخال 6 قيم كالآتي :-

```
Enter size of the array less than 50: 6
8 -5 6 -3 2 7
```

يلي ذلك عرض محتوى المصفوفة التي تشابه الآتي :-

```
The array MAT it looks like :-
MAT[0] ==>8
MAT[1] ==>-5
MAT[2] ==>6
MAT[3] ==>-3
MAT[4] ==>2
MAT[5] ==>7
```

أخيرا سنصل إلى المطلوب وهو طباعة المصفوفة other والحاوية للقيم الموجبة فقط كالآتي :-

```
While array OTHER it looks like :-
OTHER[0] ==>8
OTHER[1] ==>6
OTHER[2] ==>2
OTHER[3] ==>7
```

بالبرنامج تم استخدام جملتي for الأولى مهمتها إدخال القيم وتخزينها في المصفوفة mat والثانية لإيجاد عناصر القيم الموجبة بالمصفوفة عن طريق جملة if ومن ثم تخزينها في مصفوفة other عن طريق الدليل counter مع زيادة العدد 1 لهذا الدليل الذي يعمل كعداد لتحديد مكان تخزين القيمة الموجبة وأيضا لتحديد عدد عناصر المصفوفة other بغية استخدامه عند الطباعة .

أخيراً إذا كان شرط جملة if صحيحا أي أن قيمة العداد counter تساوي 0 فإننا نقول: إن المصفوفة الأصلية لا تحتوي قيما موجبة وبالتالي وجب الخروج عن طريق الدالة exit(0) وإيقاف البرنامج ، أما إذا كان الشرط غير

ذلك ، فعندها تأتي جملة for التي مهمتها طباعة قيم المصفوفـة other وإيقـاف البرنامج .

يمكن ضم جملتي for في جملة for واحدة كالآتي :-

```
for(i = 0 ; i < n ; i++)
{
    scanf("%d", &mat[i]);
    printf("\n\tMAT[%d] = %d", i, mat[i]);
    if(mat[i] > 0)
    {
      other[counter] = mat[i];
      counter++;
    }
}
```

مثال (11-1-7)

المطلوب كتابة برنامج مهمته القيام بقراءة قيم حقيقية وتخزينها في مصفوفة أحادية مع ترتيب هذه القيم ترتيبا تصاعديا .

```
#include <stdio.h>
#include <process.h>
#define MAX 10
main()
{
    float temp, a[MAX];
    int i, j, k, n, n1, n2, counter = 1;
    printf("\nGive number of values: ");
    scanf("%d", &n);
    if( n > 10 || n <= 0 )
      exit(0);
    for(i = 1 ; i <= n ; i++)
    {
      printf("A[%d] = ", i);
      scanf("%f", &a[i]);
    }
    n1 = n-1;
    printf("\n\nSteps of sorting array :-\n");
```

```
    for(i = 1 ; i <= n1 ; i++)
    {
      n2 = i+1;
      for(j = n2 ; j <= n ; j++)
        if( a[j] - a[i]  < 0 )
        {
          temp = a[i];
          a[i] = a[j];
          a[j] = temp;
          printf("\n STEP %d ==> ", counter);
          for(k = 1 ; k <= n ; k++)
               printf("%.1f  ", a[k]);
          counter++;
        }
    }
    printf("\n\nThe sorted array as follwing:\n");
    for(i = 1 ; i <= n ; i++)
      printf("\nA[%d] ==> %.1f", i, a[i]);
}
```

عند تنفيذ هذا البرنامج وإدخال القيم وليكن عددها 6 كالآتي :-

```
Give number of values: 6
A[1] = 2.5
A[2] = 9.1
A[3] = 6.6
A[4] = 2.5
A[5] = 7.4
A[6] = 3.0
```

سيتم عرض حالة المصفوفة في كل خطوة من خطوات التبديل كما يلي :-

```
Steps of sorting array :-

STEP 1 ==> 2.5  6.6  9.1  2.5  7.4  3.0
STEP 2 ==> 2.5  2.5  9.1  6.6  7.4  3.0
STEP 3 ==> 2.5  2.5  6.6  9.1  7.4  3.0
STEP 4 ==> 2.5  2.5  3.0  9.1  7.4  6.6
STEP 5 ==> 2.5  2.5  3.0  7.4  9.1  6.6
STEP 6 ==> 2.5  2.5  3.0  6.6  9.1  7.4
STEP 7 ==> 2.5  2.5  3.0  6.6  7.4  9.1
```

أخيرا طباعة المصفوفة بالترتيب التصاعدي لعناصرها كالآتي :-

```
The sorted array as following:

A[1] ==> 2.5
A[2] ==> 2.5
A[3] ==> 3.0
A[4] ==> 6.6
A[5] ==> 7.4
A[6] ==> 9.1
```

مثال (8-1-11)

البرنامج الآتي مهمته :

1) قراءة بيانات تخص بضاعة بالمخزن لا تزيد كميتها عن 100 وكذلك رقمها وثمنها مع تخزين هذه البيانات في مصفوفة .

2) إدخال أي رقم يمثل رقم البضاعة وبالتالي البحث عنه بالمخزن وفي حالة وجوده يطبع الرقم وثمن البضاعة ، اما في حالة عدم وجوده تطبع رسالة مناسبة تدل على ذلك .

```
#include <stdio.h>
#define LIMIT   100
main()
{
    int I, size;
    int j, found, item_no, item[LIMIT];
    float item_price, price[LIMIT];
    printf("\nEnter size of items less than 100 : ");
    scanf("%d", &size);
    for(i = 0 ; i < size ; i++)
    {
      printf("\nEnter item number [%d] ==> ", i+1);
      scanf("%d", &item[i]);
      printf("Enter price       [%d] ==> ", i+1);
      scanf("%f", &price[i]);
    }
    printf("\nEnter item number to be ");
    printf("searched (0 to QUIT): ");
```

```
      scanf("%d", &item_no);
      while( item_no != 0 )
      {
        j = 0;
        found = 0;
        do
        {
          if( item_no == item[j] )
            found = 1;
          else
            j++;
        } while(( j<size) && ( found !=1 ));
        if( found )
        {
          printf("\nThe item number %d ", item_no);
          printf("has the price %.2f", price[j]);
        }
        else
          printf("\nSorry item %d no found.",item_no);
        printf("\nEnter item number to be ");
        printf("searched (0 to QUIT): ");
        scanf("%d", &item_no);
      }
}
```

هنا وفي حالة وجود بيانات كثيرة تم استخدام مصفوفتين الأولى item من النوع الصحيح لتخزين رقم البضاعة بالمخزن ، والثانية price من النوع الحقيقي لتخزين ثمن البضاعة ، وبعد إدخال عدد 3 أصناف مثلا وحفظها بالمصفوفتين كالآتي :-

```
Enter size of items less than 100 : 3

Enter item number    [1]: 207
Enter price          [1]: 67.25

Enter item number    [2]: 835
Enter price          [2]: 10.99

Enter item number    [3]: 321
Enter price          [3]: 759.75
```

جاء السؤال عن إدخال الرقم المطلوب البحث عنه أو إدخال القيمة 0 لإيقاف البرنامج ، وفي حالة إدخال الرقم 207 بعد الرسالة كما يلي :-

Enter item number to be searched (0 to QUIT): 207

عندها يأتي الرد بالرسالة التالية :-

The item number 207 has the price 67.25

والدالة على أن ثمن البضاعة التي رقمها 207 هي القيمة 67.25 ، ويتكرر ظهور رسالة إدخال الرقم أو إيقاف البرنامج ، وفي حالة إدخال الرقم 312 بدلا من الرقم الموجود وهو 321 كالآتي :-

Enter item number to be searched (0 to QUIT): 312

عندها تظهر الرسالة التالية :-

Sorry item number 312 not found.

على شاشة العرض التي تقول بأن هذا الرقم غير موجود بالمخزن .

أما إذا تم إدخال الرقم الصحيح 321 بعد الرسالة التالية :-

Enter item nymber to be searched (0 to QUIT): 321

فسيكون الرد وجود هذا الرقم كالآتي :-

The item number 321 has the price 759.75

وأخيراً لإيقاف البحث يتم إدخال القيمة 0 كالآتي :-

Enter item number to be searched (0 to QUIT): 0

2) المصفوفة الحرفية *Characters Array*

في بعض الفصول الماضية سبق أن استعملنا المتغير الحرفي الذي له الشكل التالي :-

```
char   a;
```

وهو يعني أن المتغير الحرفي a قابل لأن يخزن فيه حرف واحد فقط في المرة الواحدة ، أما الآن فيمكن الإعلان عن متغير واحد من النوع الحرفي بحيث يمكن تحديد حجمه في ذاكرة الحاسب .

مثال (11-1-9)

الإعلان

```
char  sample[10];
```

يقصد به حجز مكان في الذاكرة لـ 10 أماكن لمتغير sample من النوع الحرفي ، مع الأخذ في الاعتبار هنا أن المترجم (Compiler) يضيف في نهاية هذه المصفوفة رمز النهاية ('\0') وعليه عند الإعلان عن مصفوفة من هذا النوع يجب أن يترك مكان بالمصفوفة لهذا الرمز .

مثال (11-1-10)

يمكن الإشارة إلى أي قيمة في المصفوفة الحرفية باستخدام الدليل (index) والبرنامج التالي يوضح هذا .

```
#include <stdio.h>
main()
{
    char name[15];
    printf("\nEnter string : ");
    gets(name);
    name[0] = 'N';
    name[4] = 'G';
    name[12] = 'D';
    name[13] = '?';
    printf("\nThe string now is : ");
    printf("%s", name);
}
```

عند تنفيذ البرنامج وإدخال الحروف التالية :-

Enter string : HOW DO YOU DO

سنتحصل على السطر الآتي :-

The string now is : NOW GO YOU DD?

الذي يجب أن نلاحظه في البرنامج السابق أن المصفوفة الحرفية name حجز لها 15 مكانا في الذاكرة بما فيه رمز النهاية ('\0') حيث يبدأ دليل المصفوفة من 0 إلى حجم المصفوفة 2- .

مثال (11-1-11)

كما ذكرنا سابقا أن المترجم يضيف الرمز ('\0') في نهاية المصفوفة ، البرنامج التالي يستفيد من ذلك ليقرأ سلسلة حرفية ثم يقوم بتحديد وطباعة طولها بداية من 0 إلى رمز النهاية .

```
#include <stdio.h>
main()
{
    char str[50];
    int n, counter = 0;
    printf("\nType your string : ");
    gets(str);
    for(n = 0 ; str[n] != '\0' ; n++)
       counter++;
    printf("The string length is %d", counter);
}
```

فيما يلي التنفيذ مع الإدخال والإخراج :

Type your string : THIS IS A TEST.
The string length is 15

بعد قراءة السلسلة عن طريق الدالة gets ، جاءت جملة for وفيها تتكرر زيادة القيمة 1 للمتغير counter في كل مرة لم يتحقق فيها شرط جملة for

وتتوقف هذه الزيادة عند الوصول إلى الرمز ('\0') الذي يدل على نهاية السلسلة الحرفية وبالتالي يتم طباعة طولها .

2.11) المصفوفات ذات البعدين Tow-d*imensional* Arrays

1) المصفوفة العددية *Array* Numbers

هي التي تتكون من مجموعة من الصفوف والأعمدة وقد تحتوي على بيانات من النوع الصحيح int أو الحقيقي float .

الصيغة العامة :

```
type   array_name [size1] [size2] ;
```

حيث :

array_name هو اسم المصفوفة .

size1 الدليل الأول يشير إلى عدد الصفوف بالمصفوفة (rows)

size2 الدليل الثانى يشير إلى عدد الأعمدة بالمصفوفة (columns)

وعليه يمكن الوصول إلى أي عنصر بالمصفوفة باستخدام اسمها مع الأدلة المحصورة بين القوسين [] .

مثال (1-2-11)

الإعلان

```
int a[3] [4] ;
```

يعني أن المصفوفة a هي من النوع الصحيح وتحتوي على 3 صفوف و 4 أعمدة أي أن بها 12 عنصرا (3x4) أول هذه العناصر هو a[0][0] وآخرها a[2][3] وهي تشبه الموضحة فيما يلي :-

العمود	الرابع	الثالث	الثاني	الأول
الصف الأول	a03	a02	a01	a00
الصف الثاني	a13	a12	a11	a10
الصف الثالث	a23	a22	a21	a20

حيث يتراوح حجمها بالنسبة للصفوف من 0 إلى 2 والأعمدة من 0 إلى 3.

مثال (11-2-2)

البرنامج الآتي يقوم بقراءة عدد من القيم ثم تخزينها في مصفوفة ذات بعدين مع طباعة هذه القيم على هيئة مصفوفة .

```
#include <stdio.h>
#include <conio.h>
#define ROW  3  /* number of rows */
#define COL  4  /* number of columens */
main()
{
    int a[ROW] [COL];
    int i, j;
    clrscr();
    printf("\nEnter elements of the array :\n");
    for(i = 0 ; i < ROW ; i++)
      for(j = 0 ; j < COL ; j++)
      {
        printf("A[%d,%d] ==> ", i, j);
        scanf("%d", &a[i][j]);
      }
    printf("\nThe array looks like :\n\n");
    for(i = 0 ; i < ROW ; i++)
    {
      for(j = 0 ; j < COL ; j++)
        printf("%d\t", a[i][j]);
      printf("\n\n");
    }
    getch();
}
```

بعد تنفيذ هذا البرنامج يكون إدخال البيانات مشابها للآتي :-

```
Enter elements of the array :
A[0,0] ==> -3
A[0,1] ==> 5
A[0,2] ==> 1
A[0,3] ==> 12
A[1,0] ==> -4
A[1,1] ==> 6
A[1,2] ==> 2
A[1,3] ==> 11
A[2,0] ==> -5
A[2,1] ==> 7
A[2,2] ==> 3
A[2,3] ==> 10
```

والإخراج هي المصفوفة التالية :-

```
The array looks like :

-3   5   1   12
-4   6   2   11
-5   7   3   10
```

في هذا البرنامج تم :-

1) تحديد عدد الصفوف ROW وعدد الاعمدة COL بالمصفوفة .

2) الإعلان عن مصفوفة a ذات البعدين من النوع الصحيح .

تلا ذلك استخدام جملة for المتداخلة بقصد قراءة القيم وتخزينها بالمصفوفة a باتجاه الصف (row wise) أي الصف الأول ثم الثاني فالثالث ، وبنفس الطريقة عند طباعة هذه المصفوفة .

2) المصفوفة الحرفية *Characters Array*

بالمثل كما في المصفوفة العددية يمكن إنشاء مصفوفة ذات بعدين لتخزين الحرفيات .

مثال (11-2-3)

الإعلان التالي :-

```
char   list_name[50][20];
```

يحجز مصفوفة داخل الذاكرة تتكون من 50 سلسلة حرفية كل واحدة منها لا يزيد طولها عن 20 حرفا ، حيث يمكن الوصول إلى أي عنصر من عناصر المصفوفة بتحديد الدليل الأول فقط ، فمثلا للوصول إلى مؤشر الدليل الثامن يمكن أن تكتب بالطريقة :

```
gets( list_name[7] );
```

أو بالطريقة :

```
gets( list_name[7] [0] );
```

مثال (11-2-4)

إدخال مجموعة من الأسماء وتخزينها في مصفوفة حرفية ذات بعدين ، ثم طباعتها ، هي مهمة هذا البرنامج .

```
#include <stdio.h>
#define  MAX 5
#define  LEN 20
main()
{
    int m, n;
    char name[MAX][LEN];
    printf("\nType %d names please :\n", MAX);
    for(m = 0 ; m < MAX ; m++)
      gets( name[m] );
    printf("\nThe names are :\n");
    for(m = 0 ; m < MAX ; m++)
    {
      n = 0;
      while( name[m][n] )
      {
        printf("%c", name[m][n]);
        n++;
```

```
        }
        printf("\n");
    }
}
```

وفيه مصفوفة حرفية ذات بعدين name تحتوي على 5 أسماء كل اسم ينبغي ألا يتعدي 15 حرفا ولإدخال الأسماء تم استخدام المتغير m الناتج من جملة for كدليل أيسر للمصفوفة name[m] الذى يمثل ترتيب الأسماء مع إهمال طول الاسم .

كذلك استعملت جملة

```
while( name[m][n] )
```

داخل جملة for وهي تعني بينما شرط while صحيح اطبع الاسم name[m][n] الذي ترتيبه m وطوله n من الحروف ، أي أن المتغير m يمثل الاسم الأول بالمصفوفة والمتغير n يزداد بالقيمة 1 في كل مرة لطبع الاسم حرفا حرفا حتي نهايته .

عند تنفيذ البرنامج وإعطاء الأسماء التالية :-

```
Type 5 names please :

KADIJHA AHMED
ENAS MOHAMMED
AYAH BASHIR
FADI SAAD
MAHA A. JALAL
```

سيطبع البرنامج الآتي :-

```
The names are :

KADIJHA AHMED
ENAS MOHAMMED
AYAH BASHIR
FADI SAAD
MAHA A. JALAL
```

مثال (11-2-5)

اكتب برنامجا كاملا يقوم بقراءة اسم الطالب وأرقام عدد 3 مواد دراسية مع الدرجات التي تحصل عليها كل طالب في كل مادة وذلك لفصل يوجد به عدد لا يزيد عن 20 طالبا ثم طباعة :

1) جميع البيانات المدخلة السابقة .
2) متوسط درجات المواد الثلاث .
3) أكبر وأقل درجة لكل طالب مع رقم تلك المادة .
4) تقدير كل طالب المتحصل عليه بناء على متوسط درجات المواد التي درسها وذلك على النحو التالي :-

الدرجة	التقدير
85 =< الدرجة =<100	ممتاز
75 =< الدرجة <85	جيد جدا
65 =< الدرجة <75	جيد
60 =< الدرجة <65	مقبول
الدرجة <50	راسب

```
#include <stdio.h>
#include <conio.h>
#define STD  "the student"
#define  MAX   20
#define LEN    15
#define SIZE   3
main()
{
    char name[MAX][LEN];
    char code[5][10];
    int i, j, n, k, number, max, min;
    float grade[SIZE], sum, avg, max_grd, min_grd;
```

```
clrscr();
printf("\nEnter number of students: ");
scanf("%d", &number);
for(i = 0 ; i < number ; i++)
{
    sum = 0;
    printf("\nEnter %s name[%d]: ", STD, i+1);
    for(k = 0 ; k < number ; k++)
    {
     getchar();
     gets( name[k] );
    }
    printf("Enter number of courses: ");
    scanf("%d", &n);
    for(j = 0 ; j < n ; j++)
    {
       printf("\n--------------------\n");
       printf("Enter grade [%d]: ", j+1);
       scanf("%f", &grade[j]);
       sum += grade[j];
       printf("Enter code course [%d]: ", j+1);
       scanf("%s", code[j]);
    }
    printf("\n\n<<== Press any key to continue ==>>");
    getch();

    clrscr();
    printf("\n\t\t\t  Page no %d", i+1);
    printf("\n\t\t\t  =========");
    printf("\n\nThe name of student is ");
    for(k = 0 ; k < number ; k++)
      for(j = 0 ; name[k][j] ; j++)
           printf("%c", name[k][j]);
    printf("\n------ ------------------");
    for(j = 0 ; j < n ; j++)
    {
       printf("\n Code Course [%d] ==> %s" ,  j+1, code[j]);
       printf("    and grade ==> %2.2f", grade[j]);
    }
    avg = sum/j;
    printf("\n\t\t\t      Average ==> %2.2f", avg);
    max_grd = min_grd = grade[0];
    max = min = 0;
```

```
       for(j = 1 ; j < n ; j++)
       {
         if( grade[j] < min_grd )
         {
               min_grd = grade[j];
               min = j;
         }
         if( grade[j] > max_grd )
         {
     max_grd = grade[j];
               max = j;
         }
       }
       printf("\n\n%s has the ", STD);
       printf("following informations : ");
       printf("\n--------------------------------");
       printf("\n\nMax grade ==> %2.2f  ", max_grd);
       printf("    code ==> %s\n", code[max]);
       printf("Min grade ==> %2.2f  ", min_grd);
       printf("    code ==> %s\n", code[min]);
       printf("%s result is ", STD);
       if( avg >= 85 )
         printf(" EXCELANT.");
       else
         if( avg >= 75 )
               printf(" VERY GOOD.");
         else
               if( avg >= 65 )
                 printf(" GOOD.");
               else
                 if( avg >= 50 )
                   printf(" PASS.");
                 else
                   if( avg < 50 )
                         printf(" FAIL.");
       printf("\n\n <<== Press any key to continue ==>> ");
       getch();
       clrscr();
     }
 }
```

في هذا البرنامج تم الآتي :-

1) الإعلان عن المصفوفة ذات البعدين name من النوع الحرفي تستخدم لتخزين اسم الطالب الذي يجب ألا يزيد عدد حروفه عن 15 حرفا (LEN) ولعدد لا يزيد عن 20 طالبا (MAX) .

2) الإعلان عن المصفوفة ذات البعدين code من النوع الحرفي على أساس تخزين رقم المادة بطول 5 حروف أو رموز .

3) الإعلان عن المصفوفة grade من النوع الحقيقي لتخزين درجات 20 طالبا .

4) منطق البرنامج قسم إلى :-

القسم الأول : إدخال عدد الطلبة number في الفصل الدراسي .

القسم الثاني : إدخال اسم الطالب وعدد المواد المسجلة له ، وبالتالي إدخال الدرجة ورقم المادة .

القسم الثالث : إخراج البيانات السابقة مع متوسط درجات كل طالب على حدة .

القسم الرابع : إيجاد أكبر وأقل درجة والتقدير مع الطباعة .

5) تمت قراءة اسم الطالب عن طريق الدالة

```
gets(name[k]);
```

باستخدام الدليل الأول من المصفوفة الحرفية name[MAX][LEN] ليشير إلى تخزين اسم الطالب في ذلك المكان ، ونفس الشيء تم مع المصفوفة الحرفية code التي مهمتها تخزين رقم كل مادة عن طريق الدليل الأول j ، تلا ذلك الحصول على أكبر وأقل درجة مع حساب متوسط درجات كل طالب وأخيرا تحديد حالة كل طالب حسب متوسط درجاته .

عند تنفيذ البرنامج السابق ، سيكون هناك نوع من الحوار بين المستخدم والحاسب أوله المطالبة بإدخال عدد الطلبة في الفصل وليكن 2 مع اسم الطالب وعدد المواد ودرجة ورقم كل مادة كما يلي :-

```
Enter number of students: 2

Enter student name[1]: NOWFAL BASHIR
Enter number of courses: 2
--------------------------

Enter grade [1]: 75
Enter course code [1]: CS115
--------------------------

Enter grade [2]: 50
Enter course code [2]: CS200

<<== Press any key to continue ==>>
```

وينتهي الحوار بالضغط علي أي مفتاح بغية الاستمرار ، وهنا يتم تنظيف الشاشة وإخراج البيانات السابقة مع المعلومات التى تشبه الآتي :-

```
                    page no 1
                    =========

The name of student is NOWFAL BASHIR
---------------------------------------------
Code course [1] ==>CS115   and grade ==> 75
Code course [2] ==>CS200   and grade ==> 50
                           Average ==> 62.50

The student has the following informations :
------------------------------------------
Max grade ==> 75.00    code ==> CS115
Min grade ==> 50.00    code ==> CS200
The student result is PASS.

<<== Press any key to continue ==>>
```

وتبقي المعلومات على الشاشة حتي يتم الضغط على أي مفتاح للاستمرار عندها يبدأ التحاور من جديد لإدخال البيانات الخاصة بالطالب الثاني التي قد تكون كالآتي :-

```
Enter student name[2]: MOHAMED KHAIRI
Enter number of courses: 3
-------------------------

Enter grade [1]: 80
Enter course code [1]: CS100
-------------------------

Enter grade [2]: 70
Enter course code [2]: MA101

-------------------------

Enter grade [3]: 90
Enter course code [3]: CS111

<<== Press any key to continue ==>>
```

لمشاهدة المعلومات الخاصة بالطالب الثاني ، يتم الضغط على أي مفتاح حيث تظهر البيانات والمعلومات المشابهة للآتي :-

```
                page no 2
                =========

The name of student is MOHAMED KHAIRI
------------------------------------------
Code course [1] ==>CS100   and grade ==> 80
Code course [2] ==>MA100   and grade ==> 70
Code course [2] ==>CS111   and grade ==> 90
                         Average ==> 80.00

The student has the following informations :
----------------------------------------
Max grade ==> 90.00    code ==> CS111
Min grade ==> 70.00    code ==> MA101
The student result is VERY GOOD.

<<== Press any key to continue ==>>
```

وأخيرا بالضغط على أي مفتاح سيتم إيقاف تنفيذ البرنامج .

3.11) القيم المبدئية *Initial* **Values**

وهي تعني إمكانية تخصيص أو شحن قيم مبدئية لأي مصفوفة عند الإعلان عنها مباشرة بشرط أن تكون هذه القيم من نفس نوع المصفوفة وإذا لم يتم تخصيص قيم مبدئية لعناصر المصفوفة ، عندها تأخذ المصفوفة قيما عشوائية أو أصفارا .

عموما قد تأخذ المصفوفة وقيمها الشكل التالي :-

```
type  array_name[size] = {  list of values  };
```

حيث يلي اسم وحجم المصفوفة القوسان {} وبينهما قيم المصفوفة .

مثال (11-3-1)

الإشهار :

```
int  a[5] = { 3 , -5 , 6 , 2 , 7 };
```

مكافئ للآتي :-

```
int  a[5];
a[0] = 3;
a[1] = -5;
a[2] = 6;
a[3] = 2;
a[4] = 7;
```

مثال (11-3-2)

```
#include <stdio.h>
#define  M 3
main()
{
    const  mat[M] = { 10, 20, 30 };
    int i;
    for(i = 0 ; i < M ; i++)
      printf("%d   ", mat[i]);
}
```

عند تنفيذه سيطبع الآتي :-

10 20 30

هنا إذا غيرنا قيمة المتغير M لتأخذ القيمة 5 ، سيطبع القيم الثلاث السابقة مع طباعة الباقي أصفاراً .

مثال (11-3-3)

في البرنامج التالي يتم تخصيص أربع قيم من النوع الحقيقي إلى مصفوفة ذات بعد واحد list ثم إيجاد وطباعة مجموع هذه القيم .

```
#include <stdio.h>
#define  M  4
main()
{
    float sum, list[M] = {5.2, 9.0, 4.4, -3.2};
    int i;
    sum = 0.0;
    for(i = 0 ; i < M ; i++)
      sum += list[i];
    printf("\nSum of array list ==> %.2f", sum);
}
```

عند تنفيذ البرنامج سيطبع السطر التالي :-

Sum of array list ==> 15.40

وهو ناتج من عملية جمع قيم العناصر المصفوفة list مبتدءا بالعنصر الأول list[0] ومنتهيا بالعنصر الأخير list[3] .

مثال (11-3-4)

البرنامج الآتي يبين تخصيص قيم حرفية إلى مصفوفة ذات بعد واحد ومن نفس النوع .

```
#include <stdio.h>
#define LEN  9
main()
{
    char a[LEN] = { 'I', ' ', 'l', 'i', 'k', 'e', ' ', 'C', '\0' };
    printf("\nHi you know %s", a);
}
```

سيتم تخزين الحروف I like C في المصفوفة الحرفية a وطباعتها على شاشة العرض ، مع ملاحظة استخدام الرمز ('\0') ليدل على نهاية هذه الحروف .

في حالة تخصيص قيمة LEN أقل من القيمة 9 سينتج عنها خطأ ، والسبب هو أنه يجب أن يكون حجم المصفوفة مساويا لعدد الحروف المطلوب تخزينها مع إضافة مكان لرمز النهاية ، ويمكن أن يأخذ الإعلان عن المصفوفة بالمثال السابق الشكل التالي الذي يعطي نفس النتيجة .

```
char a[ ] = { 'I' , ' ' , 'l' , 'i' , 'k' , 'e' , ' ' , 'C' , '\0' };
```

هنا لم نستخدم طول المصفوفة ، بل سيقوم المترجم بحساب هذا الطول وبالتالي يريح المبرمج من هذه المهمة .

مثال (11-3-5)

في هذا البرنامج يتم شحن وطباعة المصفوفة الحرفية بأيام الأسبوع المعروفة .

```
#include <stdio.h>
main()
{
    int i;
    char week_days[7][5] = { " SAT",
                             " SUN",
                             " MON",
```

```
                " TUE",
                " WED",
                " THU",
                " FRI",
              };
   printf("\nThe week days are :\n\t\t");
   for(i = 0 ; i < 7 ; i++)
     printf("%s,", week_days[i]);
}
```

في هذا البرنامج تم استخدام مصفوفة ذات بعدين من النوع الحرفي ، البعد الأول 7 يمثل عدد أيام الأسبوع أما البعد الثاني 5 فهو لتحديد عدد حروف اليوم الواحد ، كما سبق أن ذكرنا قد يلغى البعد الأول مع المصفوفة لتأخذ الشكل التالي :-

```
week_days [] [5]
```

وفي كلتا الحالتين سيطبع البرنامج عند التنفيذ ما يلي :-

```
The week days are :
            SAT, SUN, MON, TUE, WED, THU,FRI
```

4.11) تمرينات Exercises

1) قم بكتابة برنامج يقوم بقراءة 10 قيم وتخزينها بالصفوفة mat ثم اطبع هذه المصفوفة بالعكس .

2) المطلوب كتابة برنامج لطباعة المصفوفة الثنائية التالية وذلك باستخدام القيم المبدئية ، مرة وجمل التكرار مرة أخرى ، على أن تظهر المصفوفة بالشكل الآتي:-

```
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
26 27 28 29 30
```

3) أذكر موقع القيمة 17 بالنسبة للمصفوفة matrix إذا ما ما أعطيت الإشهار التالي :-

```
int matrix[4][4]  = {
                      { 10, 11, 12, -1 }
                      { 13, 14, 15, -2 }
                      { 16, 17, 18, -3 }
                      { 19, 20, 21, -4 }
                    } ;
```

4) المطلوب كتابة برنامج كامل لقراءة مصفوفة ذات بعدين mat تحتوي على n من الصفوف والأعمدة ، أوجد حاصل ضرب عناصر القطر الرئيسي وحاصل جمع العناصر التي اسفل القطر الرئيسي.

5) اكتب برنامجا لتخزين اسمك كاملا في مصفوفة ذات بعدين .

6) اكتب برنامجا لقراءة عدد من القيم وتخزينها في مصفوفة fat لها ثلاثة صفوف وأربعة أعمد ، أوجد مجموع عناصرها وأكبر عنصر فيها .

7) باستخدام الدالة int fun(int a[], int n, int x) اكتب برنامجاً يقوم بقراءة عدد من القيم وتخزينها في مصفوفة a وقيمة x بحيث تقوم هذه الدالة بارجاع عدد تكرار قيمة x في المصفوفة a .

8) اكتب برنامجا يقرأ مصفوفتين B, A كل واحدة منهما تحتوي على 5 عناصر، وأوجد حاصل جمعهما وحاصل ضربهما على أن يكون الناتج بالشكل الآتي:-

```
A          B          A + B          A * B
-------------------------------------------------
.          .            .              .
.          .            .              .
-------------------------------------------------
```

9) فصل دراسي به 20 طالباً, المطلوب كتابة برنامج لقراءة درجات هؤلاء الطلبة وتخزينها في مصفوفة stdn مع إيجاد متوسط درجات الطلبة الناجحين وعدد الطلبة الغير ناجحين .

10) اعد كتابة البرنامج بالتمرين السابق مع تخزين الطلبة الناجحين في مصفوفه pass والطلبة الغير ناجحين في مصفوفة fail مع طباعة هاتين المصفوفتين .

11) أوجد ناتج تنفيذ البرامج الآتية:-

a)

```
#include <stdio.h>
main()
{
    int i;
    char M[] = "Look C is good language";
    for(i = 0; M[i] != '\0' ; i++)
    if ( i % 2 != 0)
    M[i] = '*';
    for(i = 0; M[i] != '\0' ; i++)
        printf("%c", M[i]);
}
```

b)

```
#include <stdio.h>
#define R 3  #define C 4
main()
{
      int M[R] [C] = { 11, -2, -5, 13, -9, 20,
                       16, -3, 17, -7, 19, 10 };
      int i, j;
      for(i = 0; i < R; i++)
      for(j = 0; j < C; j++)
      if ( M[i][j] < 12 )
        ++M[i][j];
     for(i = 0; i < R; i++)
     {
       for(j = 0; j < C; j++)
         printf("%5d", M[i][j]);
       printf("\n");
     }
}
```

12) المطلوب كتابة برنامج يقوم بقراءة مصفوفتين B , A كل واحدة منهما تحتوي على 5 عناصر مع دمج عناصر هاتين المصفوفتين وحفظ الناتج في مصفوفة C وترتيب عناصرها ترتيبا تصاعديا قبل الطباعة ، فمثلا إذا كانت A = 1, 3, 9, 4, 5 و B = 3, 6, 2, 1, 9 عليه تكون

C = 1, 1, 2, 3, 3, 4, 5, 6, 9, 9

13) فصل دراسي به 3 طلبه اكتب برنامج لقراءة ارقام القيد وثلاثة امتحانات لكل طالب وبعد تخزين هذه البيانات المطلوب إيجاد متوسط كل طالب ومتوسط كل امتحان بحيث يكون الناتج بالشكل الآتي :-

NO	ID	TEST_1	TEST_2	TEST_3	AVG
1	...	...	....	...	...
2	...	...	....	...	...
3	...	...	....	...	...
AVG		...	....	...	

12

الفصل الثاني عشر

المصفوفات والدوال والمؤشرات

1.12) المصفوفات والدوال *Arrays and* **Functions**

ذكرنا سابقا أنه عند معالجة بيانات كثيرة جدا يتحتم علينا التعامل مع المصفوفات لتسهيل عملية تخزين هذه البيانات في متغيرات قليلة وبالتالي استخدامها ومعالجتها وقت الحاجة .

2.12) مصفوفات ذات البعد الواحد والدوال

One-d*imensional Arrays and* Functions

لكي يصبح البرنامج سهل الكتابة والمتابعة والفهم يجب أن يقسم إلى عدد من الدوال الفرعية ، ومن هنا يجب أن نأخذ في الاعتبار كيفية تمرير عناصر المصفوفة إلى أي دالة ومنها باستخدام المتغيرات الخارجية . (Global Variables)

مثال (12-2-1)

المطلوب قراءة عدد من القيم الصحيحة وتخزينها في مصفوفة ذات بعد واحد ثم إعادة تخزين هذه القيم بالعكس في مصفوفة أخرى عن طريق الدالة.

```
#include <stdio.h>
#define SIZE 3
int i, a[SIZE], b[SIZE];
main()
{
    void invers_a(void);
    for(i = 0 ; i < SIZE ; i++)
```

```
    {
      printf("Enter A[%d] ==> : ", i);
      scanf("%d", &a[i]);
    }
    invers_a();
    printf("\nArray after reversed is:\n");
    for(i = 0 ; i < SIZE ; i++)
      printf("\nA[%d] ==> %d", i, b[i]);
    putchar('\n');
}

void invers_a(void)
{
    int k = SIZE-1;
    for(i = 0 ; i < SIZE ; i++)
    {
       b[i] = a[k];
       k -= 1;
    }
}
```

عند تنفيذ هذا البرنامج الآتي هو الناتج :

```
Enter A[0] ==> : 5
Enter A[1] ==> : 6
Enter A[2] ==> : 7


Array after reversed is:

A[0] ==> : 7
A[1] ==> : 6
A[2] ==> : 5
```

نلاحظ هنا أن الإعلان عن المصفوفتين b, a من النوع الصحيح حدث قبل بداية الدالة الرئيسية مما يجعلها متغيرات خارجية ، حيث يمكن التعامل معها في أي دالة من دوال الدالة الرئيسية .

عندما استدعيت الدالة ()invers_a أصبحت المصفوفة a معروفة فيها ، وبالتالي تم وضع قيمها معكوسة في المصفوفة b التي هي أيضاً معروفة في الدالة الرئيسية حيث جرت طباعتها عند الرجوع من الدالة .

أيضاً قد يحدث تمرير عناصر المصفوفة بالإعلان عنها في الدالة .

مثال (12-2-2)

خذ مثلا الدالة الرئيسية التالية تقوم باستدعاء دالتين مختلفتين .

```
#define MAX 10
main()
{

      float mat[MAX];
      read_mat(n, mat);
      prnt_mat(n, mat);
}
```

فالدالة الأولى read_mat قد تأخذ الشكل التالي :-

```
float read_mat(int k , float cat[MAX])
{
      . . .
}
```

وفيها تم الإعلان عن مصفوفة cat وهي مصفوفة من نفس نوع وطول المصفوفة بالدالة الرئيسية التي مهمتها تخزين k من القيم الحقيقية وتخزينها في هذه المصفوفة .

أما الدالة الثانية prnt_mat فقد يكون شكلها الآتي :-

```
float prnt_mat(int m , float rat[])
{
  . . .
}
```

حيث أعلن عن مصفوفة غير محددة الطول مهمتها تخزين كل العناصر التي مررت اليها من المصفوفة mat بالدالة الرئيسية ، وقد يكون مهمة هذه الدالة إخراج البيانات التي تحتويها هذه المصفوفة وهي m من العناصر .

مثال (12-2-3)

يمكن إعادة البرنامج بالمثال (11-1-7) الفصل (11) الذي مهمته ترتيب عدد من العناصر تصاعديا وذلك باستدعاء عدد من الدوال المختلفة .

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
#define MAX 10
int i,j;
main()
{
    int n; float temp, mat[MAX];
    void read_mat(int , float mat[]);
    void sort_mat(int , float mat[]);
    void prnt_mat(int , float mat[]);
    clrscr();
    printf("Give number of values less than %d : ", MAX);
    scanf("%d", &n);
    if( n > MAX || n <= 0 )
      exit(0);
    read_mat(n, mat);
    sort_mat(n, mat);
    prnt_mat(n, mat);
    getch();
}

void read_mat(int k, float cat[MAX])
{
    for(i = 0 ; i < k ; i++)
    {
      printf("mat[%d]= ", i);
      scanf("%f", &cat[i]);
    }
}
```

```
void sort_mat(int m, float rat[])
{
    float temp; int n1, n2, n3, counter = 1;
    printf("\n Steps of sorting array :-");
    n1 = m-1;
    for(i = 0 ; i < n1 ; i++)
    {
      n2 = i+1;
      for(j = n2 ; j < m ; j++)
      {
       if( rat[j] - rat[i] < 0 )
       {
         temp = rat[i];
         rat[i] = rat[j];
         rat[j] = temp;
         printf("\n STEP %d ==> ", counter);
         for(n3 = 0 ; n3 < m ; n3++)
              printf("%.1f  ", rat[n3]);
         counter++;
       }
      }
    }
}
void prnt_mat(int size, float fat[])
{
    printf("\n\nThe sorted array as following:\n ");
    for(i = 0 ; i < size ; i++)
      printf("\nmat[%d] ==> %.1f", i, fat[i]);
}
```

عند تنفيذ البرنامج وإدخال عدد القيم وليكن 6 كالآتي :-

```
Give number of values less than 10 : 6
mat[0] = 2.5
mat[1] = 9.1
mat[2] = 6.6
mat[3] = 2.5
mat[4] = 7.4
mat[5] = 3.0
```

سينتج عنها إظهار خطوات ترتيب هذه القيم كما يلي :-

```
Steps of sorting array :-

STEP 1 ==> 2.5  6.6  9.1  2.5  7.4  3.0
STEP 2 ==> 2.5  2.5  9.1  6.6  7.4  3.0
STEP 3 ==> 2.5  2.5  6.6  9.1  7.4  3.0
STEP 4 ==> 2.5  2.5  3.0  9.1  7.4  6.6
STEP 5 ==> 2.5  2.5  3.0  7.4  9.1  6.6
STEP 6 ==> 2.5  2.5  3.0  6.6  9.1  7.4
STEP 7 ==> 2.5  2.5  3.0  6.6  7.4  9.1
```

وأخيراً قيم المصفوفة وهي مرتبة كالآتي :-

```
The sorted array as following:

mat[0] ==> 2.5
mat[1] ==> 2.5
mat[2] ==> 3.0
mat[3] ==> 6.6
mat[4] ==> 7.4
mat[5] ==> 9.1
```

في بداية البرنامج تم الإعلان عن المتغيرين j, i على أساس أنهما متغيران خارجيان معروفان فى الدالة الرئيسية والدوال الأخرى ، وفي الدالة الرئيسية تم الإعلان عن المصفوفة mat ذات البعد الواحد التي حجمها MAX من العناصر تلا ذلك استدعاء الدالة الأولى read_mat ومهمتها تخزين k من القيم الحقيقية في المصفوفة cat والدالة الثانية sort_mat وهي لاستقبال جميع عناصر المصفوفة mat وتخزينها في المصفوفة rat ومن بعد ترتيب عناصرها تصاعديا حسب طولها m ، وفي الوقت نفسه طباعة خطوات هذا الترتيب ، وأخيراً تمرير هذه العناصر المرتبة إلى المصفوفة mat عند نهاية هذه الدالة ، أما الدالة الثالثة prnt_mat وفيها تم تمرير العناصر المرتبة في المصفوفة mat إلى المصفوفة المقابلة fat التي لم يحدد طولها وبالتالي طباعة قيمها .

3.12) المصفوفات ذات البعدين والدوال

Two-dimensional Arrays and Functions

لكي يتم التعامل مع بيانات كثيرة ينبغي تخزينها على هيئة مصفوفة ذات بعدين وبالتالي معالجة هذه المصفوفة وقت اللزوم .

مثال (12-3-1)

البرنامج التالي يقوم بعملية إدخال مصفوفتين ذاتي بعدين من النوع الصحيح ثم إجراء عمليتي الجمع والضرب على المصفوفتين .

نظرا لطول البرنامج ، وحتى يسهل شرحه وبالتالي فهمه ومتابعته تم تقسيمه إلى عدد من الأقسام كل قسم على حدة كالآتي :-

1) القسم الأول يمثل الدالة الرئيسية وهي كما يلي :-

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAX 10
int i,j,k,R,C;
main()
{
    int a[MAX][MAX], b[MAX][MAX], c[MAX][MAX];
    void input_mat(int a[][MAX]);
    void add_mat(int a[][MAX], int b[][MAX], int c[MAX][MAX]);
    void mul_mat(int a[][MAX], int b[][MAX], int c[MAX][MAX]);
    void write_mat(int a[][MAX]);
    clrscr();
    printf("Enter number of rows and columns :");
    scanf("%d %d", &R, &C);
    input_mat(a);
    input_mat(b);
    add_mat(a, b, c);
    printf("Addition of two arrays is :\n ");
    write_mat(c);
    if( R != C )
    {
```

```
      printf(" Sorry error data \n");
      printf(" rows and columns not equal \n");
      exit(0);
    }
    mul_mat(a, b, c);
    printf("Multipliction of two arrays :\n\n");
    write_mat(c);
    getch();
}
```

وفيها تم الإعلان عن ثلاث مصفوفات c, b, a من النوع الصحيح حجم كل واحدة منها R من الصفوف و C من الأعمدة وبعد إدخال عدد الأعمدة والصفوف تم استدعاء الدالة input_mat مرتين لقراءة المصفوفة a والمصفوفة b ، تلا ذلك مناداة الدالة add_mat لكي تجمع المصفوفتين b, a وحفظ الناتج بالمصفوفة c والرجوع إلى نقطة الاستدعاء لكي يتم استدعاء دالة الطباعة write_mat حيث تطبع محتويات المصفوفة c وقبل استدعاء دالة الضرب mul_mat يتم التأكد من أن عدد الاعمدة يجب أن يكون مساويا لعدد الصفوف حتي تكتمل عملية الضرب وبالتالي تجرى طباعة الناتج عن طريق الدالة السابقة write_mat أما في حالة عدم التساوي ، فعندها تطبع الرسالة

```
Sorry error data
rows and columns not equal
```

ويتم الخروج من البرنامج .

2) القسم الثاني يحتوي على دالة input_mat ومهمتها استقبال قيم المصفوف a أولا وقيم المصفوفة b ثانيا ، مع الأخذ في الاعتبار استدعاء هذه الدالة في كل مرة ينفذ فيها هذا البرنامج ، والدالة هي :-

```
void input_mat(int a[][MAX])
{
    static int m = 1;
    printf("Enter elements of array %d :\n", m);
    for(i = 0 ; i < R ; I++)
      for(j = 0 ; j < C ; j++)
      scanf("%d", &a[i][j]);
    m++;
}
```

3) القسم الثالث يضم دالة الجمع add_mat ومهمتها جمع المصفوفتين b, a وتخزين الناتج في المصفوفة c وهي كالآتي :-

```
void add_mat(int a[][MAX], int b[][MAX], int c[][MAX])
{
    for(i = 0 ; i < R ; i++)
     for(j = 0 ; j < C ; j++)
      c[i][j] = a[i][j] + b[i][j];
}
```

4) القسم الرابع يحتوي على دالة ضرب المصفوفتين b, a حيث تعبؤ المصفوفة c بالأصفار قبل أن تستخدم في عملية تخزين حاصل ضرب المصفوفتين ويضم الدالة التالية :-

```
void mul_mat(int a[][MAX], int b[][MAX], int c[][MAX])
{
    int k;
    for(i = 0 ; i < R ; i++)
     for(j = 0 ; j < C ; j++)
     {
       c[i][j] = 0;
       for(k = 0 ; k < R ; k++)
         c[i][j] = c[i][j] + a[i][k] * b[k][j];
     }
}
```

5) القسم الخامس والأخير يحتوي على دالة طباعة ناتج حاصل الجمع والضرب وفيه الدالة التالية :-

```
void write_mat(int c[][MAX])
{
   for(i = 0 ; i < R ; ++i)
    {
      for(j = 0 ; j < C ; ++j)
        printf("%4d", c[i][j]);
      printf("\n");
    }
   printf("\n<< Press any key to continue >>");
   getch();
}
```

أخيراً يمكن ضم هذه الأجزاء كلها ووضعها في برنامج واحد ، وعند تنفيذ هذا البرنامج يتم تنظيف الشاشة ويظهر عليها رسالة تطلب إدخال عدد الصفوف والأعمدة أي 3, 3 كالآتي:-

```
Enter number of rows and columns : 3 3
```

يلي ذلك إدخال قيم المصفوفة الأولي :

```
Enter elements of array 1 :
6 6 6
5 5 5
4 4 4
```

وقيم المصفوفة الثانية :

```
Enter elements of array 2 :
2 2 2
3 3 3
1 1 1
```

وعليه يكون الناتج كما يلي :-

```
Addition of two arrays is :
8  8  8
8  8  8
5  5  5

<< Press any key to continue >>
```

وبالضغط على أي مفتاح يظهر حاصل ضرب المصفوفتين :

```
Multipliction of two arrays :
36  36  36
30  30  30
24  24  24

<< Press any key to continue  >>
```

وبالضغط على أي مفتاح يتم الرجوع إلى البرنامج .

4.12) المصفوفة والمؤشرات Array and Pointers

في الفصول الماضية سبق أن تعاملنا مع المؤشرات الحرفية ومنها الإعلان

```
char  *a;
```

وهو يعني أن المتغير a هو مؤشر (Pointer) يشير إلى قيمة من النوع الحرفي char ، فالجملة :

```
int *ptr=55;
```

تعني أنه قد تم تخزين القيمة الصحيحة 55 في الموقع المشار إليه بالمؤشر ptr، اما ما يتعلق باستخدام المؤشرات مع المصفوفات فهو ما سوف نقوم به فيما يلي .

5.12) المصفوفة ذات البعد الواحد والمؤشرات
One-dimensional Array and Pointers

مثال (12-5-1)

الإعلان

```
int mat[10];
```

يقصد به أن المصفوفة mat هي من النوع الصحيح ، حيث يمكن أن نؤشر إلى أول عنصر فيها بطريقتين :

إما mat أو mat[0]

كما يمكن استخدام أي مؤشر ليشير إلى بداية المصفوفة ، فمثلا جملة التخصيص :

```
min=mat[2];
```

حيث استخدم الدليل 2 ليشير إلى قيمة العنصر الثالث في المصفوفة mat وبالتالي إسناد تلك القيمة إلى المتغير min وكذلك الجملة :

```
min=*(mat+2);
```

لها نفس تأثير الجملة السابقة ، حيث استخدم المؤشر الذي يشير إلى العنصر mat[0] مضافا إليه العدد 2 وهو يقابل العنصر الثالث في المصفوفة mat مع الأخذ في الاعتبار أن الرمز (*) يكون قبل الأقواس ، لأن الأقواس لها أفضلية أعلى من الرمز (*) .

مثال (12-5-2)

البرنامج الآتي مهمته قراءة عدد من القيم الصحيحة مع إيجاد متوسط القيم الزوجية وذلك باستخدام المؤشرات .

```
#include <stdio.h>
#define MAX 10
float sum = 0.0;
int i, k = 0;
main()
{
    int n, mat[MAX];
    void read_mat(int, int mat[]);
    void sum_mat(int, int mat[]);
    printf("\nEnter number of values less than %d : ", MAX);
    scanf("%d", &n);
```

```
    read_mat(n, mat);
    sum_mat(n, mat);
    if( k != 0 )
     printf("\nThe avarage of even values = %.2f", sum/k);
    return 0;
}

void read_mat(int k, int *cat)
{
    for(i = 0 ; i < k ; i++)
    {
      printf("MAT[%d] ==> ", i);
      scanf("%d", &cat[i]);
    }
}
void sum_mat(int size, int *fat)
{
    for(i = 0 ; i < size ; i++)
      if( fat[i] % 2 == 0 )
      {
        sum += fat[i];
        k++;
      }
}
```

إذا نفذ هذا البرنامج وأدخلت القيم المناسبة :

```
Enter number of values less than 10 : 5
MAT[0] ==> 4
MAT[1] ==> 8
MAT[2] ==> 2
MAT[3] ==> 4
MAT[4] ==> 5
```

فسيكون الناتج مشابها للآتي :-

```
The avarage of even values = 4.50
```

بعد الإعلان عن المصفوفة بالدالة الرئيسية جآءت الدالة read_mat حيث تم الإعلان عن متغير مؤشر cat أي تخصيص اسم المصفوفة mat بالدالة الرئيسية للمؤشر cat في هذه الدالة ليشير إلى أول عنصر في المصفوفة،

حيث تمت قراءة عدد من القيم الصحيحة وتخزينها بالمصفوفة ، بعدها وعند الرجوع تم استدعاء الدالة sum_mat التي كانت مهمتها جمع القيم الزوجية وتعدادها k وبالتالي الرجوع إلى الدالة الرئيسية وطباعة متوسط هذه القيم .

مثال (3-5-12)

في هذا البرنامج يتم إدخال مصفوفة ذات بعد واحد ثم إيجاد وطباعة أصغر عنصر ومكانه في هذه المصفوفة .

```
#include <stdio.h>
#define LEN  5
int i, *ptr;
main()
{
    int mat[LEN];
    void call_where(int mat[]);
    printf("Enter %d elements of array\n\n",LEN);
    for(i = 0 ; i < LEN ; i++)
    {
      printf("Enter MAT[%d]: ", i);
      scanf("%d", &mat[i]);
    }
    call_printf(mat);
    return 0;
}
void call_printf(int fat[])
{
    int i, min, pos = 1;
    ptr = fat;
    min = *ptr;         /* put first element in min  */
    for(i = 1 ; i < LEN ; i++)
      if( *(ptr+i) < min )
      {
         min = *(ptr+i);
         pos = i;
      }
      printf("\n The smallest element is");
      printf(" MAT[%d] = %d ", pos, min);
}
```

في خارج الدالـة الرئيسية تم الإعلان عن المتغير المؤشر ptr أما في داخلها فقد جاء الإعلان عن مصفوفة mat الأحادية من النوع الصحيح ، وبعد قراءة القيم وتخزينها بالمصفوفة استدعيت الدالة الفرعية call_where التي مهمتها استقبال قيم المصفوفة بالدالة الرئيسية وحفظها بالمصفوفة fat غير المحددة الطول ، يليها الجملة :

```
ptr = mat;
```

التي تعني اسناد عنوان المصفوفة المتمثل في العنصر mat[0] إلى المؤشر ptr ، أو أن ptr يشير إلى mat ، وبما أن المطلوب إيجاد أصغر عنصر في المصفوفة فقد جاء الأمر التالي :-

```
min =*ptr;
```

وهو يعني تخصيص قيمة أول عنصر يشير إليه المؤشر ptr إلى المتغير الصحيح min ، وعليه بدأت جملة for من القيمة 1 حيث تمت المقارنة بين ثاني عنصر بالمصفوفة مع قيمة المتغير min باستخدام جملة if بالشكل التالي:-

```
if( *(ptr+i) < min )
```

في كل مرة تزداد قيمة العداد i الذي يمثل دليل عناصر المصفوفة ثم تقارن قيمة العنصر مع المتغير min ، فإذا كانت أصغر عندها تنفذ جملتان

الأولى : تخصص هذه القيمة للمتغير min باستخدام المؤشر بالطريقة :

```
min=*(ptr+i);
```

أو بالطريقة

```
min=*(&ptr[0] + i);
```

الثانية : تخصص قيمة الدليل i للمتغير pos لتحديد مكان أصغر قيمة .

الناتج سيكون بالشكل الآتي بعد الإجابة عن الأسئلة المطلوبة .

```
Enter 5 elements of array

Enter MAT[0]= 8
Enter MAT[1]= 5
Enter MAT[2]= 1
Enter MAT[3]= 2
Enter MAT[4]= 4

The smallest element is MAT[2] = 1
```

6.12) المصفوفة ذات البعدين والمؤشرات

Two-d*imensional Array and* Pointers

استخدام المؤشرات مع المصفوفة ذات بعدين يأخذ نفس طريقة المصفوفة ذات البعد الواحد .

مثال (12-6-1)

الإعلان

```
int array1[10][10] , array2[10][10];
```

يعني ان المصفوفتين arry2,array1 هما من النوع الصحيح كل واحدة منها لها 100 عنصر ، حيث يستخدم العنوان الأساسي للمصفوفة array1 ليؤشر إلى أول عنصر وبالتالي لتحديد بداية العنصر الأول فيها بطريقتين :

اما array1 أو array1[0][0]

مثال (12-6-2)

الجملة :

```
array2 [i][j] = array1 [i][j];
```

وهي جملة التخصيص المعروفة حيث تخصص القيمة الموجودة في الصف i والعمود j من المصفوفة array1 للمصفوفة array2 بنفس الأماكن .

يمكن عمل التخصيص السابق باستخدام المؤشرات بعدة طرق منها :

1) `array2 [i][j] = *(array1 [i] + j);`

2) `array2 [i][j] = (*(array + i))[j];`

3) `array2 [i][j] = *(&array1 [0][0] + 10 * i+j);`

حيث الرقم 10 يمثل عدد الأعمدة بالمصفوفة .

مثال (3-6-12)

البرنامج التالي مهمته القيام بقراءة عدد من القيم وتخزينها في مصفوفة ذات بعدين باتجاه الصف row-wise وإخراجها باتجاه العمود column-wise على شكل مصفوفة .

```
#include <stdio.h>
#define ROW 3
#define COL 3
int i,j;
main()
{
    int x[ROW][COL];
    void call_scanf(int x[][COL]);
    void call_printf(int x[][COL]);
    printf("\nEnter the array in row-wise\n\n");
    call_scanf(x);
    call_printf(x);
}

/* Function to print array x column-wise */
void call_printf(int y[][COL])
{
    printf("\nHere the array in column-wise\n\n");
    for(i = 0 ; i < COL ; i++)
    {
     for(j = 0 ; j < ROW ; j++)
       printf(" %d\t", (*(y+j))[i]);
     printf("\n");
```

```
    }
}

/* Function to read array x row-wise */
void call_scanf(int x[][COL])
{

    for(i = 0 ; i < ROW ; i++)
     for(j = 0 ; j < COL ; j++)
     {
       printf("Enter X[%d,%d]:", i, j);
       scanf("%d", &(*(x[i] + j)));
     }
}
```

وفيما يلي إدخال وإخراج القيم لهذا البرنامج وقت التنفيذ .

```
Enter the array in row-wise

Enter X[0,0] ==>1
Enter X[0,1] ==>2
Enter X[0,2] ==>3
Enter X[1,0] ==>4
Enter X[1,1] ==>5
Enter X[1,2] ==>6
Enter X[2,0] ==>7
Enter X[2,1] ==>8
Enter X[2,2] ==>9

Here the array in column-wise

1    4    7
2    5    8
3    6    9
```

بالبرنامج دالتان ، الأولى ()call_scanf ومهمتها استقبال القيم وتخزينها بالمصفوفة x باتجاه الصف row-wise باستخدام المؤشر عن طريق دالة الإدخال ()scanf بالشكل التالي :-

```
scanf("%d",&(*(x[i] + j)));
```

وتعني أيضاً

```
scanf(" %d",*(&x[0][0] + COL*i+j));
```

أما الدالة الثانية call_printf() فهي لطباعة عناصر المصفوفة باتجاه العمود column-wise عن طريق جملة for ودالة الإخراج printf() باستخدام المؤشر بالشكل التالي :-

```
printf(" %d\t",(*(y+j))[i]);
```

وهي تعني أيضاً

```
printf("%d\t",*(&x[0][0]+COL*j+i));
```

والمشابهة للطريقة العادية

```
printf("%d\t",x[j][i]);
```

مثال (4-6-12)

البرنامج الآتي يبين إحدى طرق التعامل بين المصفوفات بالطريقة العادية أو باستخدام المؤشرات وقت تمرير البيانات بين الدالة الرئيسية ومجموعة من الدوال الفرعية ، حيث توجد مصفوفتان: الأولى تحتوي عن درجات مقرر الحاسب الآلي لطلبة لا يزيد عددهم عن 10 طلبة ، الثانية تحتوي على رقم قيد كل طالب ، المطلوب طباعة درجات هذا الفصل تصاعديا بالنسبة لرقم القيد مع إيجاد أكبر درجة ورقم قيد الطالب المتحصل على هذه الدرجة.

الحل

حتي يسهل فهم ومتابعة البرنامج ، عليه ينبغي أن تكتب الدالة الرئيسية بالشكل التالي :-

```
#include <stdio.h>
#include <process.h>
#include <conio.h>
#define MAX 9
main()
{
    float grade[MAX];
    int n, id[MAX];
    void read_grade(int k, int id[MAX], float grade[]);
    void sort_grade(int n, int *id, float *p);
    clrscr();
    printf("\nEnter class size : ");
    scanf("%d", &n);
    read_grade(n, id, grade);
    sort_grade(n, id, grade);
}
```

وفيها تم الإعلان عن المصفوفتين: الأولى grade لتخزين درجات الطلبة والثانية id لتخزين أرقام قيدهم مع استدعاء الدالة read_grad أرقام القيد والدرجات والتي هي بالشكل التالي :-

```
void read_grade(int k, int id[], float grade[])
{
    int i;
    printf("\n Enter id and grade for %d students\n",k);
    for(i = 0 ; i < k ; I++)
      scanf("%d %f", &id[i], &grade[i]);
}
```

أخيراً الدالة sort_data ومهمتها استقبال قيم المصفوفتين grad, id التي تحتوي على n من العناصر عن طريق دليليها a , p من النوع المؤشر الأول يمثل رقم القيد والثاني يمثل الدرجات وبالتالي ترتيب هاتين المصفوفتين ترتيبا تصاعديا باستخدام جملتي for مع طباعة أكبر درجة ورقم قيد صاحبها ومتوسط الفصل وهي كالآتي :-

```
void sort_grade(int n, int *a, float *p)
{
    int i, j, temp, pos;
    float avg, temps, max_grade, sum = 0.0;
    for(i = 0 ; i < n ; i++)
     for(j = i+1 ; j < n ; j++)
     {
       if( *(a+i) >= *(a+j) )
       {
         temp = *(a+i);
         *(a+i) = *(a+j);
         *(a+j) = temp;
         temps = *(p+i);
         *(p+i) = *(p+j);
         *(p+j) = temps;
       }
     }
    max_grade =*p;
    pos = *a;
    for(i = 0 ; i < n ; i++)
     for(j = i+1 ; j < n ; j++)
     {
       if(  *(p+j) >= *(p+i) )
       {
         max_grade = *(p+j);
         pos = *(a+j);
       }
     }
    puts(" NO.   ID NO        GRADE ");
    puts("------------------------");
    for(i = 0 ; i < n ; I++)
    {
      printf("\n%d  %7d %10.2f",  i+1, *(a+i), *(p+i));
      sum += *(p+i);
    }
    avg = sum/n;
    printf("\n\nThe ID [%d] has the max grade ==> %.2f",
    pos, max_grade);
    printf("\n The class average = %.2f", avg);
    printf("\n <<== Press any key to go back ==>>");
    getch();
  }
```

عموما فإنه عند ربط الدالة الرئيسية مع بقية الدوال الفرعية عند التنفيذ ، سيحدث نوع من التحاور بين الحاسب والمستعمل له على شكل سؤال مطالبا بإدخال عدد الطلبة بالفصل وليكن 5 كالآتي :-

```
Enter class size : 5
```

عندها يطالب البرنامج بإدخال أرقام القيد والدرجات لعدد 5 طلبة كما يلي:-

```
Enter id and grade for 5 students
20001  77
20009  80
20003  60
20007  55
20004  40
```

أخيراً يتم طباعة البيانات السابقة بالترتيب التصاعدي مع رقم قيد الطالب المتحصل على أكبر درجة يتبعها متوسط الفصل كما يلي :-

```
NO.  ID NO    GRADE
-------------------------------
1    20001    77.00
2    20003    60.00
3    20004    40.00
4    20007    55.00
5    20009    80.00

The ID [20009] has the max grade ==> 80.00
 The class average = 62.40
 <<== Press any key to go back ==>>
```

مثال (12-6-5)

المطلوب إعادة كتابة البرنامج بالمثال (12-3-1) باستخدام المؤشرات .

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>    /*  for malloc function  */
#define MAX  10
int i,j,k,R,C;
main()
{
    int *a[MAX], *b[MAX], *c[MAX];
    void input_mat(int *a[MAX]);
    void add_mat(int *a[MAX], int *b[MAX], int *c[MAX]);
    void mul_mat(int *a[MAX], int *b[MAX], int *c[MAX]);
    void write_mat(int *c[MAX]);
    clrscr();
    printf("Enter number of rows and columns :");
    scanf("%d %d", &R, &C);
    for(i = 0 ; i < R ; i++)
    {
      a[i] = (int *) malloc (C * sizeof(int) );
      b[i] = (int *) malloc (C * sizeof(int) );
      c[i] = (int *) malloc (C * sizeof(int) );
    }
    input_mat(a);
    input_mat(b);
    add_mat(a, b, c);
    printf("Addition of two arrays is :\n ");
    write_mat(c);
    if( R != C )
    {
      printf("\nSorry error data \n");
      printf("row and columns not equal \n");
      exit(0);
    }
    mul_mat(a, b, c);
    printf("\nMultipliction of two arrays :\n");
    write_mat(c);
    getch();
}

void input_mat(int *a[MAX])
{
    static int m = 1;
    printf("Enter elements of array %d :\n", m);
    for(i = 0 ; i < R ; I++)
```

```
    for(j = 0 ; j < C ; j++)
      scanf("%d", (*(a + i ) + j ) );
   m++;
}
void add_mat(int *a[MAX], int *b[MAX], int *c[MAX])
{
    for(i = 0 ; i < R ; i++)
     for(j = 0 ; j < C ; j++)
       *(*(c + i) + j) = *(*(a + i) + j) + *(*(b + i) + j);
}
void mul_mat(int *a[MAX], int *b[MAX], int *c[MAX])
{
    int k;
    for(i = 0 ; i < R ; i++)
     for(j = 0 ; j < C ; j++)
     {
       *(*(c + i) + j) = 0;
       for(k = 0 ; k < R ; k++)
        *(*(c + i) + j) = *(*(c + i) + j) + *(*(a + i) + k) * *(*(b + k) + j);
     }
}

void write_mat(int *c[MAX])
{
    for(i = 0 ; i < R ; ++i)
     {
       for(j = 0 ; j < C ; ++j)
         printf("%4d", *(*(c + i) + j));
       printf("\n");
     }
    printf("\n<<== Press any key to continue ==>>");
    getch();
}
```

بعد إشهار المتغيرات c, b, a على أنها مصفوفات من نوع المؤشر جاء الأمر :

```
a[i] = (int *) malloc (C * sizeof(int) );
```

وفيه استخدمت الدالة malloc التي مهمتها تحديد عدد الأماكن المطلوب حجزها بالبايت بالذاكرة وبالتالي تخصيصها لمؤشر ، أما المؤثر sizeof

فمهمته احتساب السعة أثناء تنفيذ البرنامج بالبايت لمتغير من النوع الصحيح في الذاكرة ، فمثلا دالة الطباعة

```
printf("\ %d  %d ",sizeof(float) , sizeof(double) );
```

ينتج عنها تخصيص عدد 4 خانات بالبايت للنوع float وعدد 8 خانات للنوع double حيث وضع النوع بين القوسين ، أما في حالة استعمال اسم المتغير فلا يوضع بين القوسين ، تلا ذلك استدعاء الدالة input_mat ومهمتها استقبال قيم عناصر المصفوفة a عن طريق جملتي for وجملة الإدخال التي كانت بالشكل التالي :

```
scanf("%d", (*(a + i ) + j ));
```

حيث استخدم المؤثر (*) مع اسم المصفوفة لتخزين القيم الأولي بالصف i والعمود j بالمصفوفة a وبالتالي تم الرجوع بقيم هذه المصفوفة إلى نقطة الاستدعاء ، وبالمثل تم إدخال قيم المصفوفة b ، وهكذا جرى استعمال المؤشرات في عمليات الجمع والضرب والطباعة ، وعلى العموم فعند تنفيذ هذا البرنامج سيتولد عنه نفس النتائج المعطاة بالمثال المشار إليه سابقا .

7.12) تمرينات *Exercises*

1) قم بكتابة برنامج لقراءة عدد من القيم وتخزينها في مصفوفة ذات اربعة صفوف وخمسة اعمدة ثم استدعي دالة ترجع بمجموع عناصر العمود الثاني والخامس ودالة اخرى ترجع بحاصل ضرب عناصر الصف الاول والثالث .

2) أعد كتابة نفس البرنامج بالتمرين السابق باستخدام دالة واحدة عوضاً عن دالتين .

3) صمم برنامجا يقوم بقراءة عدد من القيم وتخزينها في مصفوفة mat لها ثلاثة صفوف وأربعة أعمده ، ويستدعي دالتين الاولى ترجع بمجموع عناصر هذه المصفوفة وثانية ترجع بأكبر عنصرها .

4) اكتب برنامج يقرأ 10 من القيم الصحيحة وتخزينها في مصفوفة ذات بعد واحد ثم استدعي دالة واحدة ترجع بعدد القيم السالبة وعدد القيم الفردية .

5) صمم برنامج يقوم باستدعاء دالة تقرأ مصفوفتين كل واحدة تحتوي على 4 عناصر، أوجد حاصل جمعهما عن طريق دالة add وحاصل ضربهما عن طريق دالة mul ودالة ثالثة print مهمتها طباعة الناتج .

6) قم بكتابة برنامجاً لقراءة درجات لعدد 10 طلبة وتخزينها في مصفوفة مع استدعا دالة لحساب الفرق بين اكبر درجة واصغر درجة بالمصفوفة .

7) عن طريق المصفوفات ، اكتب برنامج لقراءة وتخزين رقم الطالب ودرجته لعدد 10 طلبه استخدم دالة لطباعة اكبر درجة واصغر درجة مع رقم القيد.

8) اكتب برنامجا كاملا لقراءة درجات الطالب واسمه لعدد num من الطلبة ، ثم أوجد واطبع قائمة بأسماء الطلبة الذين لهم الاسم الأول ALI مع درجاتهم .

9) ما هو ناتج البرامج التالية :-

a)

```
#include <stdio.h>
#define R 3
#define C 4
int i, j, k = 0;
main()
{
    int M[R] [C] = { 10,8,-5,13,-9,20,16,-3,17,-7,18,15 };
    int use_fun(int M[][C]);
    use_fun(M);
    for(i = 0; i < R; i++)
    {
      for(j = 0; j < C; j++)
        printf("%5d", M[i][j]);
      printf("\n");
    }
    printf("K=%d", k);
}
int use_fun(int N[][C])
{
    for(i = 0; i < R; i++)
     for(j = 0; j < C; j++)
     if ( N[i][j] %2 == 0)
     {
       N[i][j] = N[i][j] / 2;
       k += N[i][j];
     }
}
```

b)

```
#include <stdio.h>
#define L  10
main()
{
```

```
    int rat[L] = {7, 1, -4, 2, 9, 7, 3. -10. 16, 5};
    float ans,use_fun(int rat[]);
        ans = use_fun(rat);
    printf("\n The avg %.2f", ans);
    return 0;
}
float use_fun(int rat[])
{
    int a, b, i, *ptr;
    float c, sum;
    sum = b = 0;
    ptr = rat;
    a = *ptr;
    for(i = 1 ; i < L ; i++)
     if( *(ptr+i) > a )
     {
        sum += *(ptr+i);
        ++b;
     }
     return (c = sum/b);
}
```

10) اكتب برنامجا رئيسا لقراءة المصفوفة *mat* ذات البعد واحد حجمها *size* لا يزيد عن 20 ثم يقوم باستدعاء الدالة بالصورة التالية :-

```
int max(int size , int mat[] )
```

بحيث ترجع بمتوسط القيم الزوجية الموجبة .

11) اكتب برنامجا يقرأ مصفوفة مربعة A واستخدم دالة ترجع بالرسالة :

The matrix is symmetric

في حالة التماثل أي A(m , n) = A(n , m) لجميع قيم n, m أو الرسالة :

The matrix is not symmetric

في حالة عدم التماثل .

12) المطلوب شرح ما الذي يفعله البرنامج التالي ثم تنفيذه وإدخال البيانات المناسبة وإيجاد الناتج .

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
main ()
{
    int i, j, *s, n, size, ma, mi, t;
    float ss = 0, avg;
    clrscr ();
    printf("Type size of array ==>");
    scanf("%d", &n);
    s = (int*)malloc(n*sizeof(int));
    printf("Ente elements of array ==>");
    for(i= 0 ; i < n ; i++)
    {
      scanf("%d", (s+i));
      ss += *(s+i);
    }
    ma = mi = *s;
    for(i= 0 ; i < n ; i++)
    {
      if( ma < *(s+i) )
      ma = *(s+i);
      if( mi > *(s+i) )
      mi = *(s+i);
    }
    printf("MA ==>%d\nMI ==>%d\n", ma, mi);
    for(i= 0 ; i < n ; i++)
    for(j= 0 ; j < n ; j++)
    {
      if( *(s+i) <= *(s+j) )
      {
        t = *(s+i);
        *(s+i) = *(s+j);
        *(s+j) = t;
      }
    }
    for(i = 0 ; i < n ; i++)
    printf("A[%d] ==>%d ", i, *(s+i) );
    avg = ss/n;
    printf("The averag ==>%.2f", avg);
    getch();
    return 0;
}
```

13) تتبع البرنامج الآتي:

```
main ()
{
     static int array [] ={15, 30, 55, 20, 40, 10};
     int m, *ptr;
     ptr = array;
     for( m=2 ; m<=8; m+=2)
         printf("%5d", *ptr++);
}
```

14) اكتب برنامجا لقراءة مصفوفه ذات بعدين ، ثم استخدم دالة واحدة فقط ترجع بمجموع القيم الموجبة عن طريق return ، وعدد القيم السالبة من خلال المتغير الخارجي ، وأكبر قيمة من خلال متغير من نوع المؤشر .

15) اكتب برنامجاً كاملاً لقراءة درجات فصل به 10 طلبة وتخزينها في مصفوفة مع استدعاء ثلاثة دوال ، الاولى تحسب متوسط درجات الطلبة الناجحين الثانية تحسب عدد الطلبة الغير ناجحين الثالثة لطباعة الناتج .

16) اعد كتابة البرنامج بالتمرين السابق وتخزين الطلبة الناجحين في مصفوف pass والطلبة الغير ناجحين في مصفوفة fail ثم استدعاء دالة لطباعة هاتين المصفوفتين .

17) المطلوب كتابة برنامج يقوم بقراءة مصفوفتين B , A كل واحدة منهما تحتوي على 5 عناصر مع دمج عناصر هاتين المصفوفتين وحفظ الناتج في مصفوفة C عن طريق دالة xxx وترتيب عناصرها ترتيبا تصاعديا عن طريق دالة yyy وطباعة الناتج عن طريق دالة zzz .

18) اكتب برنامجا يبحث عن قيمة X من النوع الصحيح في مصفوفة LIST ذات البعد الواحد التى طولها لا يزيد عن 20 ، فإذا كانت قيمة X موجودة فاطبع مكان وجودها ، واطبع الرسالة التالية عند عدم وجودها.

The value of X not found.

13

الفصل الثالث عشر

النوع والاتحاد والتراكيب

1.13) استخدام النوع Typedef

بالإضافة إلى إشهار مجموعة من البيانات التي تم التعرف عليها من خلال الفصول السابقة مثل الصحيح int والحقيقي float والحرفي char وغيرها ، تمكننا لغة C من استحداث أنواع جديدة لتعريف البيانات عن طريق الكلمة typedef التي لا تسمح بتعريف نوع جديد من البيانات ولكن تسمح بتعريف مكافئ لنوع موجود أصلا وبالتالي السماح باستعماله فيما بعد .

الشكل العام

```
typedef  type  new_type ;
```

حيث :

typedef كلمة محجوزة مهمتها السماح بتحديد النوع الجديد .

type نوع بيانات موجود .

new_type الاسم الجديد المطلوب استخدامه .

مثال (13-1-1)

الأمر

```
typedef  int NUMBER;
```

به الاسم NUMBER كتب بحروف كبيرة حتى يكون مختلفا عند استخدامه في الإعلان وذلك بتحديد أنواع مشتقة من هذا الاسم كما يلي :-

```
NUMBER num1, num2;
```

وهذا يعني أنه تم تعريف كل من المعرفين num2, num1 على أنهما متغيران من النوع الصحيح وهذا يطابق الأمر الآتي :-

```
int num1, num2;
```

مثال (13-1-2)

ما هي ناتج تنفيذ البرنامج الآتي :-

```
#include <stdio.h>
main()
{
    typedef int NUMBER;
    typedef char *STRING;
    NUMBER sum, num1, num2;
    STRING ch;
    num1 = 10;
    num2 = 15;
    ch="\nThe sum of two numbers ==>";
    sum = num1+num2;
    printf("%s %d", ch, sum);
    return 0;
}
```

بعد الإعلان عن المتغيرات sum, num2, num1 من النوع الصحيح أسندت القيمتان 10 ,15 للمتغيرين num1, num2 على التوالي ثم جرى إيجاد مجموعهما وتخزينه بالمتغير sum الذي ينتج عنه الآتي :-

```
The sum of two numbers ==> 25
```

مثال (13-1-3)

التعامل مع السلسلة الحرفية يوضحه البرنامج الآتي .

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
typedef char STRING[80];
int use_typedef( STRING a);
main()
{
    char *str1;
    clrscr();
    printf("\nEnter string : ");
    gets(str1);
    use_typedef(str1);
    getch();
    return 0;
}

use_typedef(STRING str2)
{
    STRING str3;
    strcpy(str3, str2);
    printf("\nYour string ==> %s", str3);
    return 0;
}
```

في هذا البرنامج تم استخدام typedef في الدالة use_typedef حيث تم تحديد أنواع مشتقة من المتغير الحرفي STRING وهما المتغيران str2, str1 وبالتالي إذا نفذ هذا البرنامج وتم إدخال السلسلة سيظهر الناتج المشابه للآتي :-

```
Enter string : This is an example using typedef
Your string ==> This is an example using typedef
```

مثال (13-1-4)

باستخدام typedef اكتب برنامجا يقوم بقراءة عدد من القيم الصحيحة وتخزينها في مصفوفتين حجم كل منهما 5 عناصر مع إيجاد حاصل ضربهما.

```
#include <stdio.h>
#define MAX 5
typedef int ARRAY[MAX];
int mult(ARRAY x, ARRAY y, ARRAY z);
int i;
main()
{
    ARRAY a, b, c;
    printf("\n*** INPUT DATA PROGRAM ***\n");
    printf("\nInput %d values for array one : ", MAX);
    for(i = 0 ; i < MAX ; i++)
      scanf("    %d", &a[i]);
    printf("Input %d values for array two : ", MAX);
    for(i = 0 ; i < MAX ; i++)
      scanf("    %d", &b[i]);
    mult(a, b, c);
    printf("\nThe follwoing is data output\n");
    printf("\n array 1  array 2  array 1 * array 2");
    printf("\n-----------------------------------");
    for(i = 0 ; i < MAX ; i++)
    {
       printf("\n    %d \t    %d", a[i], b[i]);
       printf(" \t\t %d", c[i]);
    }
    printf("\n  ********** THE END **********");
    return 0;
}

int mult(ARRAY x, ARRAY y, ARRAY z)
{
    for(i = 0 ; i < MAX ; i++)
      z[i] = x[i] * y[i];
    return 0;
}
```

تنفيذ هذا البرنامج ينتج عنه إظهار الرسالة المناسبة ، وبعد إدخال القيم لكل مصفوفة على النحو التالي :-

```
*** INPUT DATA PROGRAM ***

Input 5 values for array one : 3  4  5  6  7
Input 5 values for array two : 4  5  6  7  8
```

تكون النتائج كالآتي :-

```
The follwoing is data output

array 1  array 2  array 1 * array 2
-----------------------------------
  3     4          12
  4     5          20
  5     6          30
  6     7          42
  7     8          56
********** THE END **********
```

بالبرنامج تم اشهار المتغير ARRAY كمصفوفة من النوع الصحيح طولها 5 عناصر عن طريق استخدام typedef ومن ثم استخدام هذا المتغير لإشهار المتغيرات c, b, a التي أصبحت جميعها متغيرات على شكل مصفوفات صحيحة كل واحد منها يحتوي على 5 عناصر .

بعد قراءة المصفوفتين b, a وإرسالهما إلى الدالة mult بنفس الطريقة جرى استخدام المتغير ARRAY لإشهار المتغيرات z, y, x ثم إيجاد حاصل ضرب المصفوفة x في المصفوف y وتخزين الناتج في المصفوفة z وأخيراً الرجوع بهذا الناتج إلى الدالة الرئيسية وطباعته هناك .

2.13) الاتحاد Union

هو إمكانية تخزين ومشاركة أكثر من عضو أو متغير ذى أنواع مختلفة في نفس المكان والعنوان بذاكرة الحاسب ، أي تمكين المبرمج من استغلال سعة الذاكرة الاستغلال الأمثل حيث يقوم المترجم (Compiler) بحجز مكان

واحد لأكثر من متغير ، وفي حالة تخصيص قيمة لأي من هـذه المتغيرات ، فإن القيم الموجودة سابقا تلغى عند تنفيذ البرنامج .

وقد يأخذ الاتحاد الشكل :-

```
union  union_name
{
     type  member_1;
     type member_2;
     . . .
     type member_n;
}union_variable;
```

أو الشكل :-

```
union  union_name
{
     type  member_1;
     type member_2;
     . . .
     type member_n;
};
union_name  union_variable;
```

حيث :

union_name اسم الاتحاد أو التجمع ويستخدم لتعريف متغـيرات أخرى في البرنامج .

member العضو أو العنصر وهي المحصورة بين قوسي الفئة {} .

union_variable اسم المتغير من النوع الموحد الذى يأخذ أكبر حجم من الذاكرة بحسب الأعضاء المكونة له .

يمكن الوصول إلى أي عضو أو عنصر من عناصر الاتحاد عن طريق اسم الاتحاد union_name يتبعه مؤثر النقطة (.) ثم العضو أو العنصر المعني member ، أي:

```
union_name.member;
```

مثال (13-2-1)

البرنامج التالي مهمته إسناد قيم مختلفة إلى مجموعة من العناصر تحت اسم موحد .

```
#include <stdio.h>
main()
{
    union example
    {
      int x;
      char y;
    }sample;
    sample.x = 66;
    sample.y = 'A';
    printf("\nX = %d in decimal", sample.x);
    printf("\nWhile Y = %c in character", sample.y);
}
```

الذي يحدث عند تنفيذ هذا البرنامج هو الآتي :-

```
X = 65 in decimal
While Y = A in character
```

حيث تم الإعلان عن الاتحاد الذي يضم عضوين هما y, x من النوع الصحيح والحرفي في بداية الدالة الرئيسية تحت اسم موحد هو sample ، ثم خصصت القيم A, 66 للعنصرين y, x على التوالي ، وعند التنفيذ نلاحظ أن العنصر x أصبح له القيمة 65 غير القيمة المسندة إليه أصلا وهي 66 ، والسبب أن كلا من العنصرين y, x يشاركان فى نفس المكان والقيمة المدخلة أخيراً وهي 65 المقابلة للحرف A في نظام آسكي (ascii) .

مثال (13-2-2)

مخزن به عدد من الأصناف ، المطلوب كتابة برنامج لإدخال رقم الصنف ومعدل مبيعات هذه الأصناف في كل شهر من الشهور الثلاثة الأولى من السنة.

```
#include <stdio.h>
#include <conio.h>
#define SIZE   4
#define MONTH  3
int i, k;
union target
   {
     int a[5];
     int item_number[MONTH];
   }item;

main()
{
    union target sale[SIZE];
    void call_fun();
    int call_calculation(union target sales[SIZE]);
    clrscr();
    printf("Please enter data for the following");
    printf("\n*********************************\n");
    printf("ITEM NO MONTH1 MONTH2 MONTH3\n");
    for(k = 0 ; k < SIZE ; k++)
    {
      scanf("%d", &item.item_number[k]);
      for(i = 0 ; i < MONTH ; i++)
        scanf("%d", &sale[k].a[i]);
    }
    call_fun();
    call_calculation(sale);
}

int call_calculation(union target sales[SIZE])
{
    int total[SIZE] = {0, 0, 0, 0};
    for(i = 0 ; i < SIZE ; i++)
      for(k = 0 ; k < MONTH ; k++)
        total[i] = total[i] + sales[i].a[k];
    for(k = 0 ; k < SIZE ; k++)
    {
      printf("  %d  ", item.item_number[k]);
      for(i = 0 ; i < MONTH ; i++)
        printf("%5d", sales[k].a[i]);
      printf("%5d\n", total[k]);
    }
```

```
}

void call_fun()
{
    printf("\nNow data with total amount");
    printf(" of sales as the following\n");
    printf("\t    *******************\n");
    printf(" ITEM NO. MONTH1  MONTH2");
    printf(" MONTH3  TOTAL\n");
    printf("---------------------------------\n");
}
```

وهذه هي البيانات المدخلة والنتائج المقابلة لها وقت تنفيذ البرنامج .

```
Please enter data for the following
*********************************
ITEM NO MONTH1 MONTH2 MONTH3
1111    15    12    20
2222    10    14    18
3333    16    15    17
4444    20    22    25

Now data with total amount of sales as the following
*********************************************
ITEM NO. MONTH1  MONTH2  MONTH3  TOTAL
-------------------------------------
1111     15     12     20    47
2222     10     14     18    42
3333     16     15     17    48
4444     20     22     25    67
```

الملاحظ في هذا البرنامج أنه قد تم استخدام union مع المتغير item الذي يضم عضوين من النوع الصحيح هما المتغير a والمتغير item_number وعليه أصبح المتغير item_numer هو أحد أعضاء المتغير item ومن ثم يمكن التعامل معه بإدخال رقم الصنف عن طريق الجملة التالية :-

```
scanf("%d",&item.item_number[k]);
```

حيث k يمثل عدد الأصناف في المخزن .

أما بخصوص العضو الآخر a فهو يعتبر عنصرا من عناصر المتغير الموحد sale الذي هو على شكل مصفوفة لاستعمالها في إدخال عدد الأصناف المباعة في كل شهر من الشهور الثلاثة عن طريق الجملة التالية:-

```
scanf("%d", &sale[k].a[i]);
```

حيث المتغير i يمثل عدد الأصناف المباعة في كل شهر التي عددها k صنف .

3.13) التراكيب *Structures*

التركيبة تعني إمكانية تجميع مجموعة من المتغيرات تسمي بالعناصر أو الأعضاء بحيث تكون من نفس النوع أو من أنواع مختلفة كل واحدة منها لها خانة أو حقل (Field) تحت اسم واحد بحيث يمكن الرجوع إلى هذه المتغيرات عن طريق اسم التركيبة ومعالجتها كوحدة واحدة .

ويتم الإعلان عن التركيبة بالشكل :

```
struct  struct_name
{
     type  member_1;
     type member_2;
     . . .
     type member_n;
}struct_variable;
```

أو الشكل :-

```
struct  struct_name
{
     type  member_1;
     type member_2;
     . . .
     type member_n;
};
struct  struct _name   struct _variable;
```

حيث :

struct كلمة محجوزة تعتبر إشهارا للتركيبة .

struct_name اسم التركيبة وهو اختياري .

memebr اسم العضو أو العنصر والمحصورة بين قوسي الفئة {} .

struct_variable اسم المتغير من نوع التركيبة .

مثال (13-3-1)

التركيبة :

```
struct  book
{
    char title[25];
    char author[30];
    char publisher[25];
    float price;
    int year;
}my_book;
```

تحتوي على خمسة عناصر تحت اسم book وهي :-

1) العناصر publisher, author, title متغيرات من النوع الحرفي لعنوان ومؤلف وناشر الكتاب .

2) العنصر price متغير من النوع الحقيقي يمثل ثمن الكتاب .

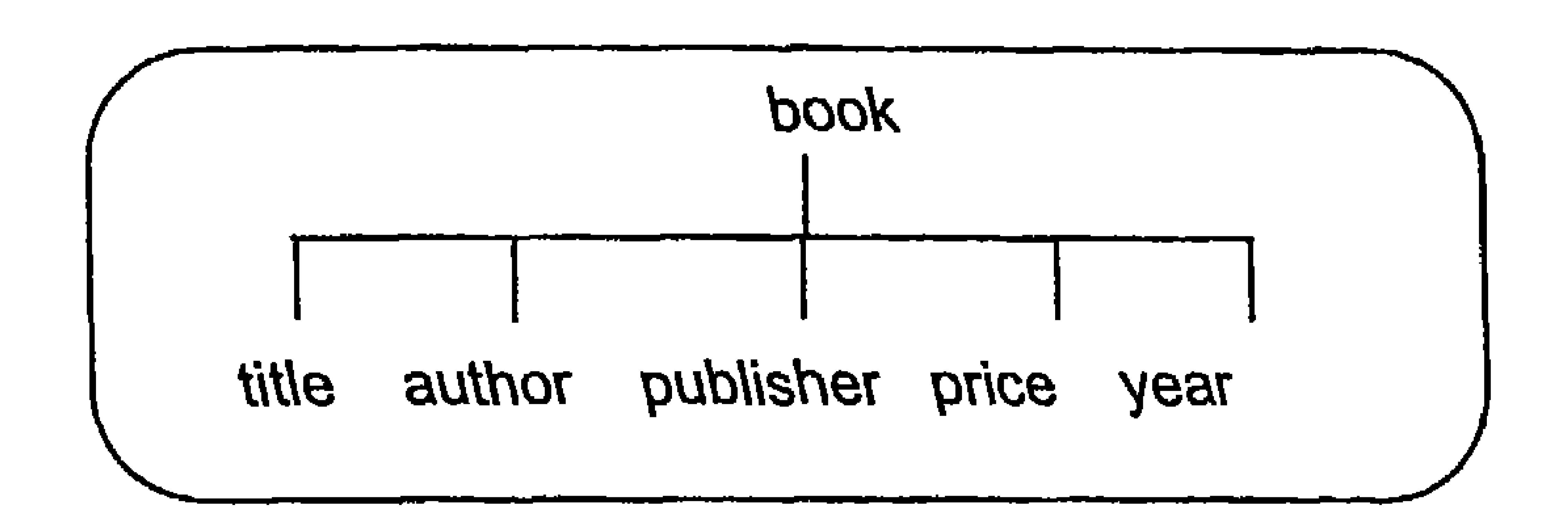
3) العنصر year متغير من النوع الصحيح يمثل سنة إصدار الكتاب .

أما my_book يعتبر اسم متغير التركيبة .

عند التعامل مع أي عنصر من عناصر التركيبة يكون كالآتي :-

```
struct_variable.member;
```

أي اسم متغير التركيبة struct_variable يتبعه مؤشر النقطة (.) ثم العنصر أو العضو المعني member ، وفيما يلي التخطيط الهيكلي لهذه التركيبة :-



مثال (13-3-2)

البرنامج التالي يتم فيه إسناد تاريخ الميلاد عن طريق التركيبة .

```
#include <stdio.h>
main()
{
    struct date
    {
      int day;
      int month;
      int year;
      char *name;
    }birthday;
    birthday.name = "MIRATH BASHIR ";
    birthday.day = 9;
    birthday.month = 4;

    birthday.year = 1994;
    printf("\nHi %s your birthday is ", birthday.name);
    printf("%d/", birthday.day);
    printf("%d/", birthday.month);
    printf("%d", birthday.year);
}
```

في هذا البرنامج تم الإعلان عن التركيبة تحت الاسم الاختيارى date وهى تحتوي على أربعة عناصر : الثلاثة الأول متغيرات من النوع الصحيح والرابع من النوع الحرفي محصورة جميعها بين قوسي الفئة يتبعها birthday وهو متغير من النوع date ، حيث تم تخصيص الاسم واليوم والشهر والسنة

لهذه المتغيرات عن طريق اسم المتغير birthday يليه مؤثر النقطة (.) يليها اسم العنصر .

عند تنفيذ هذا البرنامج سينتج عنه السطر التالي :-

Hi MIRATH BASHIR your birthday is 9/4/1994

مثال (13-3-3)

يمكن إعادة كتابة البرنامج بالمثال السابق باستخدام القيم الإبتدائية مع اسم التركيبة .

```
#include <stdio.h>
main()
{
    struct date
    {
      int day;
      int month;
      int year;
      char *name;
    }birthday = {9, 4, 1994, " MIRATH BASHIR "};
    printf("\nHi %s your birthday is ", birthday.name);
    printf("%d/", birthday.day);
    printf("%d/", birthday.month);
    printf("%d", birthday.year);
}
```

استخدم المتغير birthday مع تخصيص القيم المبدئية التي يجب أن تكون متطابقة من حيث النوع والعدد ومحصورة بين قوسي الفئة { } ، وإذا نفذ هذا البرنامج ، فسيعطي نفس الناتج في البرنامج السابق .

يمكن أيضاً استخدام typedef مع التركيبة struct وذلك بإعادة الإشهار كالآتي :

```
typedef strct
    {
       nt day;
       int month;
       int year;
       char *name;
    } date;
      date birthday;
```

وهي تعني أن المتغير birthday من نفس نـوع المتغير date الـذي يعتبر تركيبة تضم ثلاثة متغيرات .

4.13) التراكيب والدوال Structures and Functions

لتكون العلاقة واضحة بين التركيبة والدالة ، نسوق المثال التالي :-

مثال (13-4-1)

المطلوب كتابة برنامج مهمته استدعاء دالـة فرعيـة مهمتهـا قـراءة قيمتين من النوع الصحيح مع طباعتهما بالدالة الرئيسية .

```
#include <stdio.h>
struct two_values
{
   int x;
   int y;
} a, fun();

main()
{
    a = fun();
    printf("Value one ==> %d\n", a.x);
    printf("Value two ==> %d\n", a.y);
}

struct two_values fun()
{
    struct two_values temp;
```

```
    printf("Enter two integer values: ");
    scanf("%d %d", &temp.x, &temp.y);
    return temp;
}
```

هنا تم الإعلان عن التركيبة تحت اسم two_values التي تضم متغيرين y, x من النوع الصحيح ، والمتغير a والدالة fun من نوع التركيبة two_values، حيث استدعيت الدالة fun وفيها تم الإعلان عن المتغير temp من نفس نوع التركيبة المعلن عنها خارجيا وعليه فهو يضم نفس المتغيرات من حيث العدد والنوع التي تضمها التركيبة two_values ، من هنا نستطيع قراءة القيمتين وإسنادهما للعضوين y, x .

عموما فإنه عند التنفيذ سيكون الناتج مشابها للآتي :-

```
Enter two integer values: 30  66
Value one ==> 30
Value two ==> 66
```

مثال (13-4-2)

يمكن إعادة كتابة البرنامج بالمثال (13-3-2) باستخدام الدالة على النحو المبين بهذا البرنامج .

```
#include <stdio.h>
struct date
{
   int day;
   int month;
   int year;
   char *name;
} ;

main()
{
      struct date birthday;
      int print_date(date birth );
```

```
    birthday.name = "MIRATH BASHIR ";
    birthday.day = 9;
    birthday.month = 4;
    birthday.year = 1994;
    print_date(birthday);
    return 0;
}

int print_date( date ddmmyy)
{
    printf("\nHi %s your birthday is ", ddmmyy.name);
    printf("%d/", ddmmyy.day);
    printf("%d/", ddmmyy.month);
    printf("%d", ddmmyy.year);
    return 0;
}
```

هنا تم الإشهار عن التركيبة date خارج الدالة الرئيسية ولم يتم فيها استخدام اسم متغير التركيبة بين قوس الفئة المغلق } والفاصلة المنقوطة (;) تلا ذلك الإعلان عن المتغير birthday بالدالة الرئيسية من نوع التركيبة date باستخدام الكلمة struct كما يلي :-

```
struct date birthday;
```

وبالتالي إرسال كل عناصر التركيبة إلى الدالة print_date عن طريق المتغير birthday وفيها تم الإعلان عن دليلها المتغير ddmmyy من نفس نوع التركيبة birthday باستخدام اسم التركيبة date و الكلمة struct كالآتي :-

```
struct date ddmmyy;
```

حيث قامت الدالة باستقبال وإخراج البيانات المرسلة إليها من الدالة الرئيسية كما يلي :-

```
Hi MIRATH BASHIR your birthday is 9/4/1994
```

مثال (3-4-13)

البرنامج الآتي يوضح كيفية استخدام عناصر التركيبة من أنواع مختلفة.

```
#include <stdio.h>
struct rtype
{
  int n;
  char *str;
  int a[10];
  int sum;
} ;
main()
{
    int print_rtype(struct  rtype q);
    int i;
    struct rtype rec;
    rec.sum = 0;
    printf("\nEnter your string : ");
    gets(rec.str);
    printf("\nEnter number of values less than 10 : ");
    scanf("%d", &rec.n);
    printf("Enter %d integer numbers : ", rec.n);
    for(i = 0 ; i < rec.n ; i++)
    scanf("%d", &rec.a[i]);
    rec.sum = print_rtype(rec);
    printf("\nWhich has the sum = %d", rec.sum);
}

int print_rtype(struct rtype record)
{
    int i;
    printf("\n%s", record.str);
    printf(" has %d values as following :", record.n);
    for(i = 0 ; i < record.n ; i++)
    {
     printf("\nA[%d] = %d", i, record.a[i]);
     record.sum += record.a[i];
    }
    return (record.sum);
}
```

في بداية البرنامج تم الإعلان عن التركيبة تحت اسم rtype، التي تضم أربعة عناصر : المتغير n من النوع الصحيح والمتغير s من النوع الحرفي و المتغير a مصفوفة ذات بعد واحد حجمها 10 عناصر من النوع الصحيح والمتغير sum من النوع الصحيح، أيضاً جرى الإعلان عن الدالة print_rtype التي مهمتها استقبال وطباعة البيانات من نفس النوع والعدد التى تتكون منها التركيبة rec التي هي المكافئة للتركيبة rtype، بعد قراءة السلسلة الحرفية وعدد من القيم الصحيحة ، جاء استدعاء الدالة حيث تم طباعة السلسلة أولا ثم حساب مجموع قيم عناصر المصفوفة a والرجوع بها إلى نقطة الاستدعاء وطباعتها هناك .

وقت تنفيذ البرنامج وإدخال السلسلة الحرفية وعدد 5 قيم كما يلي :-

```
Enter your string : Hi user your array
Enter number of values less than 10 : 5
Enter 5 integer numbers : 10  20  30  40  50
```

سينتج الآتي :-

```
Hi user your array has 5 values as following
A[0] = 10
A[1] = 20
A[2] = 30
A[3] = 40
A[4] = 50
Which has the sum = 150
```

5.13) التراكيب والمؤشرات *Structures* and *Pointers*

توجد طريقة أخرى للتعامل مع التركيبة المعقدة أو ذات العناصر المتعددة وذلك باستخدام المؤشرات pointers بواسطة مؤشر التركيبة (<-) أي مؤثر الطرح يتبعها مؤثر أكبر من وهو يأخذ الصورة التالية :-

```
strcut_variable -> member;
```

حيث متغير اسم التركيبة strcut_variable يتبعه المؤشر <- ثم عنصر أو عضو التركيبة member مع ملاحظة ان المؤثرين (. , <-) لهما الأسبقية القصوي من بين مؤثرات لغة C .

مثال (13-5-1)

يمكن إعادة كتابة البرنامج في المثال (13-4-3) باستخدام مؤشر التركيبة على النحو التالي :-

```
#include <stdio.h>
struct rtype
{
   int n;
   char *str;
   int a[10];
   int sum;
} ;

main()
{
    int sum_of_n_numbers(struct  rtype *q);
    int i;
    struct rtype rec;
    rec.sum = sum_of_n_numbers(&rec);
    printf("\n%s", rec.str);
    printf(" has %d values as following :", rec.n);
    for(i = 0 ; i < rec.n ; i++)
      printf("\nA[%d] = %d", i+1, rec.a[i]);
    printf("\nWhich has the sum = %d", rec.sum);
    return 0;
}

int sum_of_n_numbers(struct rtype *q)
{
    int I;
    q -> sum = 0;
    printf("\nEnter your string : ");
    gets( q -> str);
    printf("\nEnter number of values < 10 : ");
```

```
    scanf("%d", &q -> n);
    printf("Enter %d integer numbers : ", q -> n);
    for(i = 0 ; i < q -> n ; i++)
    {
       scanf("%d", &q -> a[i]);
       q -> sum += q->a[i];
    }
    return (q->sum);
}
```

في هذا البرنامج استخدم المتغير q كدليل للدالة sum_of_n_numbers من النوع المؤشر وهو من نوع التركيبة rtype ، وعليه استعمل مؤشر التركيبة (<-) عند إدخال السلسلة والقيم وعددها بهذه الدالة .

وحتي يمكن التعامل مع مؤشر التركيبة ، نورد الأمر :

```
q -> c = 'X';
```

وهو يعني تخزين الحرف X في المتغير الحرفي c المشار إليه بالمؤشر q.

وبالمثل الأمر :

```
q -> n = 2;
```

مهمته تخزين القيمة 2 في المتغير الصحيح n المشار إليه بالمؤشر q .

في حين الأمر :

```
q-> a[1] = 4;
```

يعني تخزين القيمة 4 في عنصر المصفوفة a[1] المشار إليه بالمؤشر q .

يمكن إعادة الدالة sum_of_n_numbers باستخدام مؤشر (*) عوضا عن مؤشر التركيبة كما يلي :-

```
int sum_of_n_numbers(struct rtype *q)
{
    int i;
    (*q) . sum = 0;
```

```
    printf("\nEnter your string : ");
    gets( (*q).str );
    printf("\nEnter number of values <= 10 : ");
    scanf("%d", &(*q).n);
    printf("Enter %d integer numbers : ", (*q).n);
    for(i = 0 ; i < (*q).n ; i++)
    {
      scanf("%d", &(*q).a[i]);
      (*q).sum += (*q).a[i];
     }
     return ((*q) . sum);
}
```

هنا وجب استخدام الأقواس لأن مؤثر النقطة (.) له الأسبقية على مؤثر (*)، وفيما يلي بعض جمل الإسناد باستعمال مؤثر (*) وهي :

```
(*q).c = 'X';
(*q).n = 2;
(*q).a[1[ = 4;
```

الأن يمكن تعريف التركيبة كمصفوفة ذات بعد واحد أو بعدين على حد سواء ، فالتركيبة :

```
struct array
{
    int num1;
    int num2;
}matrix1, matrix2;
struct array matrix1[30];
```

بها المصفوفة matrix1 تحتوي على 30 عنصرا كل واحد منها عبارة عن تركيبة تضم عنصرين num2, num1 وعليه يمكن إسناد قيمة صحيحة 77 إلى العنصر num1 بالتركيبة وذلك بالعنصر الخامس من المصفوفة matrix1 على النحو التالي :-

```
matrix1[4].num1 = 77;
```

في حين التركيبة :

```
struct array
{
    float mat[10][10];
    float value1;
}matrix;
```

عن طريقها يمكن استخدام أمر التخصيص التالي :-

```
matrix.mat[3][4] = 12.34;
```

الذي يعني إسناد وتخزين القيمة الحقيقية 12.34 بالصف الرابع العمود الخامس من المصفوفة mat .

مثال (13-5-2)

المطلوب كتابة برنامج كامل لإدخال رقم الطالب واسمه وعدد ثلاثة امتحانات لكل طالب وذلك لفصل دارسي به عدد NUM من الطلبة مع عمل الآتي:-

1) طباعة كل البيانات المدخلة السابقة مع مجموع درجات الامتحانات الثلاثة لكل طالب .
2) حساب متوسط كل امتحان بالفصل .
3) حساب أكبر درجة لكل امتحان .

```
#include <stdio.h>
#include <conio.h>
#define N_ST  2
#define TESTS  3
int i, j;
struct student
{
    char name[20];
    int idno;
    float exam[TESTS];
} ;

main()
{
```

```
float avg[TESTS], total_exam[TESTS], max[TESTS];
struct student *std[N_ST];
float total(student *std[N_ST], float total_exam[N_ST]);
float avg_exam(student *std[N_ST], float avg[N_ST]);
float max_grade(student *std[N_ST], float max[TESTS]);
clrscr();
for(i = 0 ; i < N_ST ; i++)
{
  printf("Enter idno[%d] ==> ", i+1);
  scanf("%d", &std[i] -> idno);
  printf("Enter name[%d] ==> ", i+1);
  getchar();
  gets( std[i] -> name );
  for(j = 0 ; j < TESTS ; j++)
  {
    printf("Enter exam[%d] ==> ", j+1);
    scanf("%f", &std[i] -> exam[j]);
  }
  printf("\n");
}
total(std,total_exam);
avg_exam(std,avg);
max_grade(std,max);
printf("\n-----------------------------------\n");
printf(" ID NO   NAME   TEST1");
printf("  TEST2   TEST3   TOTAL\n");
printf("-------------------------------------\n");
for(i = 0 ; i < N_ST ; i++)
{
  printf(" %d\t%s", std[i] -> idno, std[i] -> name);
  for(j = 0 ; j < TESTS ; j++)
    printf("  %.2f", std[i] -> exam[j]);
  printf("  %.2f\n", total_exam[i]);
}
printf("-----------------------------------\n\n");
for(i = 0 ; i < TESTS ; i++)
{
  printf("The average of test %d", i+1);
  printf(" is %.2f\n", avg[i]);
}
printf("\n\n");
for(i = 0 ; i < TESTS ; i++)
{
```

```
        printf("The maxamum grade in test ");
        printf("%d is %.2f\n", i+1, max[i]);
      }
      getch();
      return 0;
}

float total(struct student *std[N_ST], float total_exam[N_ST])
{
      float sum;
      for(i = 0 ; i < N_ST ; i++)
      {
        sum = 0.0;
        for(j = 0 ; j < TESTS ; j++)
          sum += std[i] -> exam[j];
        total_exam[i] = sum;
       }
}

float avg_exam(struct student *std[N_ST], float avg[N_ST])
{
      float sum;
      for(j = 0 ; j < TESTS ; j++)
      {
        sum = 0.0;
        for(i = 0 ; i < N_ST ; i++)
          sum += std[i] -> exam[j];
        avg[j] = sum/N_ST;
      }
}

float max_grade(struct student *std[N_ST], float max[TESTS])
{
      for(i = 0 ; i < TESTS ; i++)
      {
        max[i] = std[0] -> exam[i];
        for(j = 1 ; j < N_ST ; j++)
        {
          if( std[j] -> exam[i] > max[i] )
            max[i] = std[j] -> exam[i];
        }
      }
}
```

عند تنفيذ هذا البرنامج وإدخال البيانات التي تخص الطالب الأول :

```
Enter idno[1] ==> 1234
Enter name[2] ==> NADIA SALEM
Enter exam[1] ==> 75
Enter exam[1] ==> 62
Enter exam[1] ==> 69
```

والبيانات التى تخص االطالب الثاني :

```
Enter idno[1] ==> 9876
Enter name[2] ==> ALI BASHIR
Enter exam[1] ==> 73
Enter exam[1] ==> 65
Enter exam[1] ==> 71
```

تكون المخرجات كالآتي :-

```
-------------------------------------------
ID NO    NAME    TEST1  TEST2  TEST3   TOTAL
-------------------------------------------
1234   NADIA SALEM  75.00  62.00  69.00  206.00
9876   ALI BASHIR   73.00  65.00  71.00  209.00
---------------------------------------------
The average of test 1 is 74.00
The average of test 2 is 63.50
The average of test 3 is 70.00

The maxamum grade in test 1 is 75.00
The maxamum grade in test 2 is 65.00
The maxamum grade in test 3 is 71.00
```

مثال (13-5-3)

يمكن إشهار متغير تحت نفس الاسم في أكثر من تركيبة ، خذ مثلا النموذج الآتي:-

```
struct one
{
    char ch;
    int a;
    float b;
} ;
```

```
struct two
{
    char ch;
    int a;
} ;
struct one first;
       two second;
```

نلاحظ أن المتغير ch هو أحد العناصر في التركيبتين one ,two وعليه يمكن الإشارة إلى هذا العنصر في التركيبة two كالآتي :-

```
second.ch
```

وأيضا يمكن الإشارة إلى المتغير a بالتركيبة first :

```
first.a
```

أو التركيبة second :

```
second.a
```

بدون أي إرباك.

مثال (13-5-4)

خذ مثالا آخر

```
struct address
{
    char name[30];
    int street;
    char city[25];
    long phone;
} ;
```

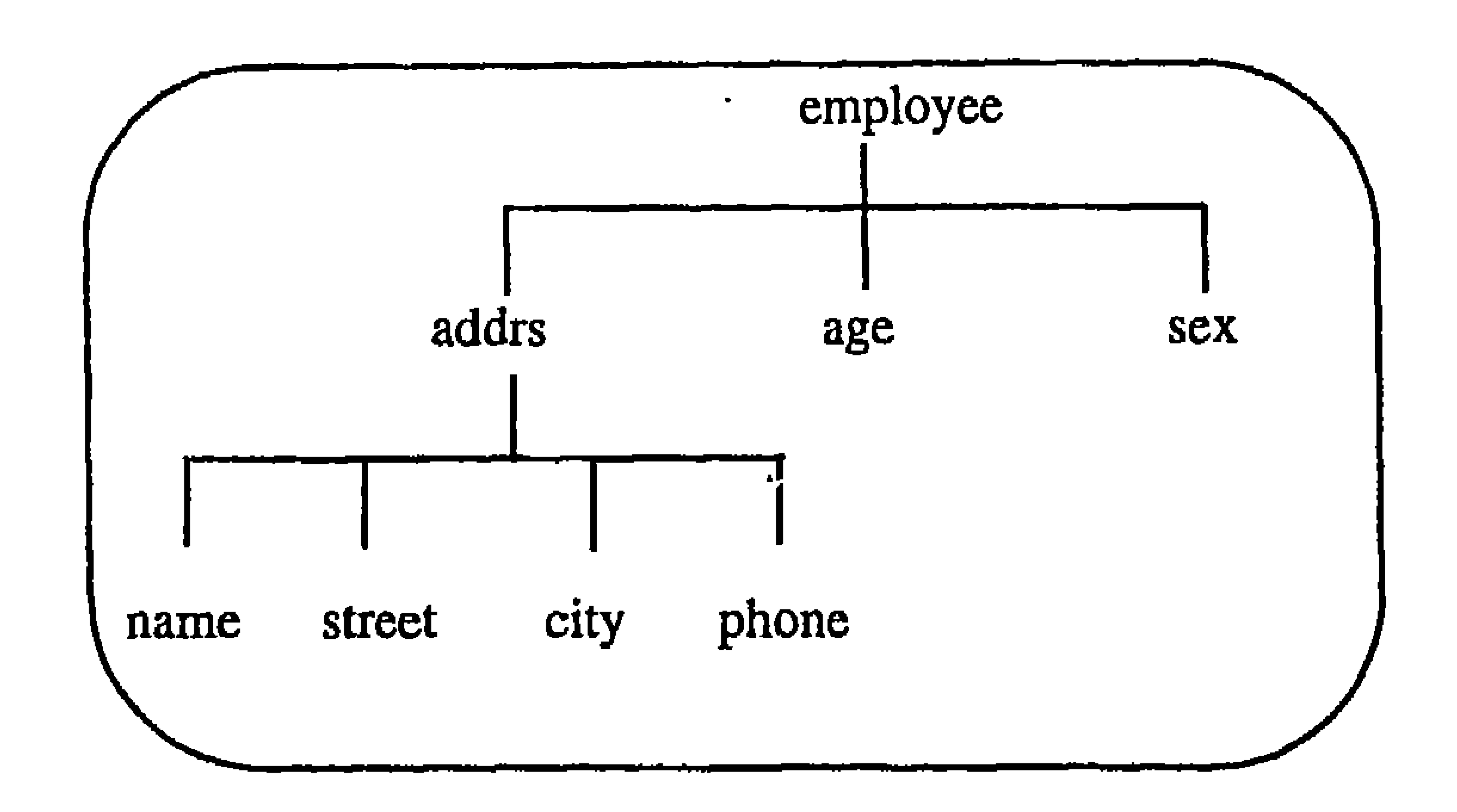
هنا نستطيع استخدام نفس تركيبة address في إشهار تركيبة أخرى تحت اسم موظف employee كما يلي :-

```
struct employee
{
    struct address addrs;
    float age;
    char sex;
} worker;
```

وعلى ضوء ذلك يجوز إسناد أي عضو كالاسم أو الشارع أو المدينة أو رقم الهاتف أو العمر أو الجنس إلى متغير التركيبة worker ، فعلى سبيل المثال لتخصيص الاسم ALI نكتب الأمر بالصورة التالية :-

```
worker.addrs.name = "ALI";
```

وحتي تكون الصورة أوضح عند استعمال هذا النوع من التراكيب ، نبين فيما يلي التخطيط الهيكلي للتركيبة (موظف employee).



مثال (13-5-5)

البرنامج التالي يتم فيه إدماج التركيبة structure مع النوع typedef حيث يتم إدخال اسم الصنف وثمنه وتاريخ شرائه ثم إصدار قيمة الفاتورة الكلية لعدد NUM من الأصناف المختلفة .

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define NUM  2
main()
{
    typedef struct
    {
      int day;
```

```
    int month;
    int year;
  } date;
  struct item
  {
    char name[20];
    char price[8];
    date dmy;
  } ;
  struct item store[ITEM_NUM];
  int i; float sum = 0.0;
  clrscr();
  for(i = 0 ; i < NUM ; i++)
  {
     printf("Enter item name ==> ");
     scanf("%s", &store[i].name);
     printf("Enter item price ==> ");
     scanf("%s", &store[i].price);
     printf("Enter day ==> ");
     scanf("%d", &store[i].dmy.day);
     printf("Enter month ==> ");
     scanf("%d", &store[i].dmy.month);
     printf("Enter year ==> ");
     scanf("%d", &store[i].dmy.year);
     clrscr();
   }
  for(i = 0 ; i < NUM ; i++)
    sum += atof(store[i].price);
  printf("------------------------------\n");
  printf("NO  ITEM   ITEM   INVOICE\n");
  printf("    NAME   PRICE   DATE\n");
  printf("------------------------------\n");
  for(i = 0 ; i < NUM ; i++)
  {
    printf("  %d\t   %s   ", i+1, store[i].name);
    printf("%s   %d/ ", store[i].price, store[i].dmy.day);
    printf("%d/%d\n", store[i].dmy.month, store[i].dmy.year);
  }
  printf("------------------------------\n");
  printf("\n\tTotal price = %.2f\n", sum);
  return 0;
}
```

في هذا البرنامج تم الإعلان عن التركيبة date التي تضم ثلاثة عناصر اليوم day والشهر month والسنة year ثم استعملت هذه التركيبة في التركيبة item وبالتالي فإن التركيبة date أصبحت أحد عناصر التركيبة item التي بدورها تضم 5 عناصر ، أيضاً تم الإعلان عن مصفوفة store[NUM] كبنية التي حجمها NUM من الأصناف ، فمثلا يمكن الإشارة إلى البيانات الخاصة بالصنف الأول كما يلي :-

1) الاسم store[0].name
2) الثمن store[0].price
3) اليوم store[0].dmy.day
4) الشهر store[0].dmy.month
5) السنة store[0].dmy.year

أما الجملة :

```
sum += atof(store[i].price);
```

ففيها استعملت الدالة atof() التي سبق شرحها ومهمتها تبديل الحرف char في نظام آسكي (ascii) إلى عدد حقيقي .

أخيراً عند تنفيذ هذا البرنامج سيتم تنظيف شاشة العرض ثم إدخال البيانات التي تخص الصنف الأول وقد تكون كالآتي :-

```
Enter item name ==> BOOK
Enter item price ==> 15.75
Enter day ==> 12
Enter month ==> 7
Enter year ==> 2000
```

بعدها يتم تنظيف الشاشة مرة أخرى والمطالبة بإدخال البيانات للصنف الثاني ويتكرر هذا حتي تنتهي عملية الإدخال لبقية الأصناف الأخرى ، وفي حالة إدخال بيانات لعدد ثلاثة أصناف ، تكون النتائج مشابهة للآتي :-

```
------------------------------------------
NO  ITEM    ITEM    INVOICE
    NAME    PRICE   DATE
------------------------------------------
1   BOOK    90.75   12/7/2000
2   PEN     66.50   21/7/2000
3   PENCIL  30.25   25/7/2000
------------------------------------------

Total price =187.50
```

6.13) تمرينات Exercises

1) أذكر الفرق الأساسي بين:

a)

```
union
structure
```

b)

```
scanf("%d", &ptr.mat[2]);
scanf("%d", &(*ptr).mat[2]);
```

2) المطلوب تصميم تركيبة تحت اسم بيانات شخصية Personal Data تضم الآتي :-

* الاسم Name
* العنوان Address : المنزل Home ، العمل Office
* رقم الهاتف Tel no : المنزل Home ، العمل Office
* رقم التلكس Telex No ، رقم البطاقة Identity Card No ، رقم جواز السفر Passport No
* رخصة القيادة Driving License ، لوحة السيارة Car Plate

3) اكتب برنامجا لتصميم تركيبة تخص فاتورة استهلاك الكهرباء تضم اسم المستهلك ، رقم العداد ، العنوان ، قيمة الاستهلاك ، وتركيبة داخلية تخص نوع الاستهلاك وتضم مساكن ، ورش ، مزارع ، محلات تجارية، تم اطبع التالي :-

* فاتورة تضم كل البيانات الخاصةِ بالمزارعين مع المجموع الكلي لقيمة الاستهلاك .
* رقم العداد واسم المستهلك لأكبر قيمة استهلاك خاصة بأصحاب المساكن .

* قائمة تضم كل البيانات السابقة لأصحاب الورش والمحلات التجارية .

4) اكتب التركيبة المناسبة تحت اسم الدواء (drug) تضم الحقول التالية :-

* اسم الدواء name من النوع الحرفي .
* رقم جهة إنتاج الدواء no من النوع الصحيح .
* العنوان address الذى يضم الحقول التالية :-
  - الشارع street من النوع الحرفي .
  - المدينة city من النوع الحرفي .
  - تاريخ انتهاء الصلاحية date_end ويضم الحقول اليوم day والشهر month والسنة year .
* الكمية quantity من النوع الصحيح .
* السعر price من النوع الحقيقي .

5) باستخدام typedef صمم تركيبة تخص مركز الشرطة تضم العناصر التالية:-

* اسم السائق Driver name .
* رقم الرخصة License Number .
* عمر السائق Driver Age .
* الجنس Sex .
* نوع الرخصة License Type وتضم أولي ، ثانية ، ثالثة .
* تاريخ الإصدار License Date .

وذلك لعدد N من السائقين مع استخدام الدوال للحصول على :

* طباعة تقرير يضم أرقام الرخص مع أسماء كل السائقين الحاملين للرخص من الدرجة الثانية الذين تقل أعمارهم عن40 سنة .

* طباعة تقرير آخر به كل أسماء الذكور والحاملين للرخص من الدرجة الثالثة مع تاريخ الإصدار.

6) صمم تركيبة لمعالجة البيانات الخاصة بنزلاء فندق معين ، حيث يتم قراءة الاسم الساكن ، رقم الحجرة ، تكلفة المبيت باليوم الواحد ، عدد الأيام . ثم اكتب دالة مهمتها طباعة أسماء كل الساكنين بالفندق مع المجموع الكلي للتكلفة ودالة أخرى تطبع رقم الحجرة حسب التكلفة .

7) اكتب برنامجا به التركيبة baby_data التي تضم اسم المولود مكان الولادة، الجنس ، التاريخ الذي يضم العناصر: اليوم ، الشهر ، السنة مع استدعاء دالتين:

* الأولى تطبع أسماء الذكور مع العناوين بترتيب تصاعدي حسب الأسماء.
* الثانية تطبع قائمة تضم أسماء الاناث وتاريخ ولادتهن مع العنوان .

14

الفصل الرابع عشر

الملفات

1.14) تعريف الملف *File Definition*

حتى الآن ، وفي كثير من البرامج التي كتبت سابقا ، تم إدخال البيانات عن طريق لوحة المفاتيح وبالتالي معالجتها وتخزينها في ملفات مؤقته (Temporary files) بذاكرة الحاسب أو إخراجها على شاشة العرض ، وبهذه الطريقة تكون هذه الملفات أقل نفعا لأنها بمجرد الانتهاء من تنفيذ البرنامج تفقد كل البيانات المدونة فيها .

عليه يمكن استخدام الملفات دائمة التخزين (Permanent files) في أوساط عديدة مثل الأقراص والأشرطة حيث تمتاز هذه الملفات بالرجوع إليها وقت الحاجة .

قبل الدخول في معالجة الملفات ، يجب علينا تعريف الملف، الذى هو عبارة عن تركيبة بيانية تضم العديد من السجلات (Records) حيث يتكون كل سجل من مجموعة من الخانات أو الحقول (Fields) مثل اسم وعنوان ورقم هاتف أى موظف ، حيث كل حقل يتكون من مجموعة من الحروف أو الأرقام أو الرموز أو جميعها .

2.14) إنشاء الملف Creating a File

قبل التعامل مع الملف ، وجب اتباع الآتي :-

1) استخدام ملف العناوين <stdio.h> الذى عن طريقه يتم معالجة الملفات.

2) تحديد قناة التعامل مع الملف بإشهاره للمعالج (Compiler) وذلك عن طريق التركيبة التي تسمي FILE ومهمتها وصف الملف .

الشكل العام :

```
FILE *pointer_name;
```

حيث pointer_name متغير من نوع المؤشر الخاص ليشير إلى FILE الذي يستخدم لتخزين البيانات والمعلومات في الملف .

3) فتح الملف باستخدام الدالة fopen وشكلها :

```
fopen( filename , mode);
```

حيث filename اسم الملف المراد معالجته .

و mode تحديد صيغة فتح الملف ، والجدول الآتي يبين كيفية التعامل مع mode :-

meaning	mode
للقراءة فقط من ملف نصوصي	r
للكتابة في ملف نصوصي مع الغاء البيانات السابقة	w
للإضافة في نهاية ملف نصوصي موجود مع عدم الغاء البيانات السابقة	a
للقراءة فقط من ملف ثنائي	rb
للكتابة في ملف ثنائي	wb
للإضافة إلى ملف ثنائي	ab
للقرءاة والكتابة في ملف نصوصي	r+
للقراءة والكتابة مثل (r+) مع الفرق إنشاء سجل جديد وإلغاء القديم	w+
مفتوح للقراءة والكتابة للإضافة عند نهاية الملف الثنائي	a+
للقراءة والكتابة في ملف ثنائي	rb+
للقراءة والكتابة مثل (rb+) مع الفرق إنشاء سجل جديد والغاء القديم	wb+
مفتوح للقراءة والكتابة للإضافة عند نهاية الملف الثنائي	ab+

حيث يشير المؤشر إلى بداية الملف في حالة الحرف r أو الحرف w ونهايته في حالة الحرف a ، مع الأخذ في الاعتبار إمكانية كتابة (rb+) بالشكل r+b .

4) إغلاق الملف الذي تم فتحه عن طريق الدالة fclose وشكلها :

```
fclose( pointer_name );
```

3.14) الملفات النصية Text Files

وهي تتكون من أسطر متعددة لبيانات من النوع الحرفي كل سطر له نهاية محددة لتمييزه عن بقية السطور ، حتى يمكن قراءتها والتعرف عليها من قبل الشخص المستخدم وينتهي الملف بالعلامة feof التي تدل على نهايته.

1) الكتابة في الملف Writing in a File

مثال (14-3-1)

لمعرفة كيفية إنشاء وتخزين البيانات ، البرنامج الآتي يوضح الطريقة .

```
#include <stdio.h>
#include <conio.h>
#define NUM 5
main()
{
    FILE *stream;
    int i;
    long num;
    float grade;
    char course_cod[15];
    clrscr();
    stream = fopen("PROG1.DAT" , "w");
    printf("Enter id number, grade and course code");
    printf(" for %d students : \n",NUM);
    for(i = 1 ; i <= NUM ; i++)
    {
      scanf("%ld", &num);
```

```
      scanf("%f", &grade);
      scanf("%s", course_cod);
      fprintf(stream,"%ld  %f  %s\n", num, grade, course_cod);
    }
    fprintf(stream,"\n");
    fclose(stream);
  }
```

تم الإشهار عن متغير stream من نوع المؤشر ليشير إلى التركيبة FILE التى يجب كتابتها بالحروف الكبيرة كما هي معرّفة في الملف <stdio.h> وبالتالي فإن هذا المتغير سوف يستخدم عند معالجة البيانات في الملف .

أما الجملة :

```
stream = fopen("PROG1.DAT" , "w");
```

فهي تعني أنه تم فتح ملف تحت الاسم PROG1.DAT باستخدام الحرف w الذي وضع بين علامتي التنصيص المزدوجتين ، وعليه فإن هذا الملف قد أعِدّ للكتابة عليه فقط وإن وجد هذا الملف سابقا فسوف يتم إلغاء بياناته .

باستخدام جملة for تم إدخال عدد خمسة سجلات كل سجل يضم رقم الطالب num من النوع الصحيح ودرجته grade من النوع الحقيقي ورقم المقرر course_cod من النوع الحرفي وبالتالي كتابة هذه السجلات على الملف المشار إليه بمؤشر الملف stream عن طريق الدالة fprintf() التي تأخذ الشكل التالي :-

```
fprintf(pointer_name , format , data);
```

حيث :

pointer_name اسم المتغير المؤشر من نوع التركيبة FILE .

format وصف أو تشكيل البيانات التي ستكتب في الملف .

data البيانات المراد كتابتها في الملف .

أخيراً تم إغلاق الملف عن طريق الدالة fclose الذي فتح في البداية .

عموما فإنه وقت تنفيذ هذا البرنامج ستظهر الرسالة الاتية :-

Enter id number, grade and course code for 5 students :

التي تطالب بإدخال رقم الطالب ودرجته ورقم المقرر لعدد 5 طلبة ، وإذا تحقق ذلك بإدخال البيانات التالية :-

```
90100  80.0  CS115
90105  75.0  MA200
90111  60.0  CS207
90124  84.0  EL102
90125  50.0  AR101
```

سيتم حفظها في الملف PROG1.DAT على هيئة خمسة سجلات .

وللتأكد من أن البرنامج قد أنشأ ملفا جديدا تحت اسم PROG1.DAT يمكن استعمال أمر التشغيل DIR من النظام DOS أو الأمر TYPE لفحص محتويات هذا الملف .

2) الإضافة إلى الملف *Appending to a File*

في بعض الأحيان يتحتم على المبرمج إضافة بيانات قد تم نسيانها أو بيانات جديدة إلى ملف سبق إنشاؤه والكتابة فيه .

مثال (14-3-2)

على فرض أننا نريد إضافة السجلات الثلاثة التالية :-

```
90200  62.0  ST101
90233  78.00  CS331
90250  90  CS450
```

إلى السجلات الموجودة بالملف PROG1.DAT الذي أُنشأ بالمثال (14-3-1) ، هنا علينا تغيير دالة فتح الملف بالبرنامج السابق لتأخذ الشكل التالي :-

```
stream=fopen("PROG1.DAT" , "a");
```

حيث استُخدِم الحرف a للدلالة على إضافة وكتابة هذه البيانات في نهاية الملف ، وبالتالي فان الملف يحتوي على 8 سجلات .

3) قراءة الملف Reading a File

مثال (14-3-3)

حتى يتم شرح كيفية قراءة البيانات ، البرنامج الآتي مهمته استدعاء الملف PROG1.DAT مع قراءة البيانات الموجودة عليه وإظهارها على شاشة العرض .

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
main()
{
    FILE *stream;
    int i; long id;
    float grade;
    char course_cod[15];
    clrscr();
    stream = fopen("PROG1.DAT" , "r");
    if( stream == NULL )
    {
      printf("Error file not found ");
      exit(0);
    }
    printf("\n id number   grade  course code\n");
    printf("\n --------------------------\n");
    while ( !feof(stream) )
    {
      fscanf(stream,"%ld  %f  %s", &id, &grade, &course_cod);
      printf("%ld   %.2f   %s\n", id, grade, course_cod);
    }
    fclose(stream);
}
```

بعد إشهار مؤشر الملف stream وفتحه عن طريق الجملة الآتية:-

```
stream=fopen("PROG1.DAT" , "r");
```

التي تعني أن اسم الملف PROG1.DAT من النوع النصي وقد أُعِدَّ للقـراءة منه فقط حيث استخدم الحرف r في هذه الحالة .

جآءت جملة if التالية :-

```
if( stream == NULL)
```

وتعني أنه إذا كانت عملية فتح الملف PROG1.DAT ناجحة، فسوف ترجـع الدالة ()fopen بمؤشر إلى النوع المشار إليه بواسطة stream ، أما إذا كان غير ذلك ، عندها ترجع نفس الدالة بالمؤشر الصفري NULL أو القيمة (0) ، حيث NULL كلمة معرّفة في الملف <stdio.h> وبالتالي تطبع الرسالة :

```
Error file not found
```

والخروج نهائيا من البرنامج عن طريق الدالة exit(0) .

يمكن كتابة استدعاء الدالة fopen وتعيين قيمة المؤشر إلـى stream والتـأكد من أن قيمة stream تساوي NULL في جملة واحدة كالآتي :

```
if( (stream = fopen("PROG1.DAT" , "r") ) == NULL )
```

التي تعتبر نموذجا جيدا للبرمجة .

وحيث إن الملف قد تم إنشـاؤه عـن طريـق البرنـامج فـي المثـال(14-3-1) وأضيفت إليه بعض السجلات بالمثال (14-3-2) عليه سيتم تنفيـذ جملـة while التالية :-

```
while ( !feof(stream) )
```

التي تستخدم بهذه الصورة في حالة عدم معرفتنـا بعـدد السـجلات بـالملف PROG1.DAT وهي تفيد أنه بينما لم يشر مؤشر الملف stream إلـى النهايـة ،

نفذ الجمل التالية لجملة while أي قراءة البيانات بالملف باستخدام الدالة fscanf() وطباعتها على الشاشة كما يلي :-

```
id number  grade    course code
------------------------------------
90100      80.00    CS115
90105      75.00    MA200
90111      60.00    CS207
90124      84.00    EL102
90125      50.00    AR101
90200      62.00    ST101
90233      78.00    CS331
90250      90.00    CS450
```

مثال (4-3-14)

اكتب برنامجا كاملا لإدخال مجموعة من القيم من النوع الصحيح ، أوقف عملية الإدخال عندما يتم إدخال أي حرف أو رمز ، مع حفظ هذه القيم في ملف نصي ومن ثم قراءة وطباعة هذه القيم غير المحددة العدد مع إيجاد حاصل جمعها .

```
#include <stdio.h>
#include <conio.h>
main()
{
    FILE *fpt;
    int num, sum = 0;
    char file_name[20];
    clrscr();
    printf("Enter file name please ==> ");
    gets(file_name);
    fpt = fopen(file_name, "w");
    printf("\nInput integer values or");
    printf("\nany character to stop ==>");
    while( scanf("%d", &num) )
      fprintf(fpt,"%3d", num);
    fclose(fpt);
    fpt = fopen(file_name,"r");
    printf("The values in file look like ==> ");
```

```
    while( fscanf(fpt,"%d ",&num) != EOF )
    {
      printf("%3d", num);
      if( num>0 )
        sum += num;
    }
    printf("\n\nThe sum of positive values = %d", sum);
    fclose(fpt);
}
```

بهذا البرنامج يمكن للمبرمج اختيـار اسم الملـف المقبـول بـدلا مـن الاسـم الثابت عن طريق المتغير الحرفي file_name ، أما الجملة :

```
while( scanf("%d", &num) )
```

فهي تعني قراءة الرقم المدخـل مـن النـوع الصحيـح وتخزينـه فـي الملـف المعني وتكرار ذلك حتي يتم إدخال قيمة غـير عدديـة ، عندهـا يجـب إغـلاق الملف الذي تم فتحه .

وحيث إن الملف أغلق ، عليه يجب فتحه مرة أخرى لغرض القـراءة منـه وقد استخدمت جملة while
مرة ثانية بالشكل :

```
while( fscanf(fpt,"%d ",&num) != EOF )
```

لغرض قراءة البيانات من الملف وطباعتها وإيجاد مجموعها، ويتكرر هـذا حتى الوصول إلى نهاية الملف حيث تنتهي جملة while ويطبع المجموع.

عند تنفيذ هذا البرنامج ، سيكون إدخال الملف أولا ويليه إدخال القيـم ثانيـا وأخيرا الناتج كما يلي :-

```
Enter file name please ==>sum.dat

Input integer values or
any character to stop ==> 3 4 8 2 -5 7 X
The values in file look like ==> 3 4 8 2 -5 7

The sum of positive values = 24
```

مثال (14-3-5)

البرنامج التالي يوضح أسلوب قراءة البيانات من ملف سبق إنشاؤه حيث يحتوى عددا غير معروف من الأسطر ، وبالتالي المطلوب إيجاد الآتي :-

1) عدد الفراغات .

2) عدد الحروف الهجائية .

3) عدد الأرقام .

4) عدد الأسطر .

وحفظها في ملف آخر .

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
main()
{
    int blank, letter, digit, line;
    char in_data[20], out_data[20];
    FILE *in_file , *out_file;
    clrscr();
    printf("Enter data file name ==> ");
    scanf("%s", in_data);
    printf("\nEnter output data file ==> ");
    scanf("%s", out_data);
    in_file = fopen(in_data, "r");
    if( in_file == NULL )
    {
        printf("\n [%s] can not be found", in_data);
        getch();  exit(0);
    }
    out_file = fopen(out_data, "w");
    if( out_file == NULL )
    {
        printf("\nEerror file [%s] not created ", out_file);
        getch();   exit(0);
    }
    blank = letter = digit = line = 0;
    while( !feof(in_file) )
    {
```

```
        ch = fgetc(in_file);
        if( ch == '\n' )  line += 1;
         if( ch == ' ' )  blank += 1;
          if( (ch>='a' && ch <='z') || (ch >='A' && ch <='Z' ) )  letter += 1;
               if( ch >='0' && ch <='9' ) digit += 1;
    }
    fclose(in_file);
    fprintf(out_file,"\n Here is [%s] ", out_data);
    fprintf(out_file,"\n ****************\n" );
    fprintf(out_file,"\n  1) Number of blanks ==>");
    fprintf(out_file," %d", blank);
    fprintf(out_file,"\n  2) Number of letters ==>");
    fprintf(out_file," %d", letter);
    fprintf(out_file,"\n  3) Number of digits ==>");
    fprintf(out_file," %d", digit);
    fprintf(out_file,"\n  4) Number of lines ==>");
    fprintf(out_file," %d", line);
    fclose(out_file);
}
```

عند تنفيذ هذا البرنامج ، سوف يتم الآتي :-

1) ظهور الرسالة :

```
Enter data file name ==>
```

طالبة طباعة اسم الملف الموجود بـه البيانـات المـراد معالجتهـا ، وعليه تمت طباعة الملف تحت اسم input.dat وهو الذي سبق تخزينـه وهو الآتي :-

```
Hi user
This program written on mon 9/9/2000
finds number of blanks, letters,
digits and lines.
```

2) في حالة وجوده يرد الحاسب بالرسالة الآتية:

```
Enter output data file ==>
```

طالبا اسم الملف المطلوب أن تدون فيه نتيجة معالجة البيانات ، حيث جرى اختيار الملف تحت اسم output.dat

3) استعملت جملة while :

```
while( !feof(in_file) )
```

وهي تفيد قراءة البيانات من الملف المذكور حرفا حرفا عن طريق الدالة fgetc( ) وبالتالي حساب المطاليب المذكورة في هذا المثال مادام لم يشر مؤشر الملف in_file إلى النهاية .

4) وحتي تكون الصورة واضحة ، تم كتابة بعض التوضيحات مع النتائج على ملف output.dat آخر خاص بالمخرجات .

عموما فإنه بعد إدخال اسم الملف الذي يحتوى على البيانات المطلوب معالجتها كما يلي :-

```
Enter output data file ==> input.dat
```

ويليه إدخال اسم الملف الجديد الذي تخزن فيه المعلومات كما يلي :-

```
Enter data file name ==>output.dat
```

بعدها يتم تخزين كل المطاليب بهذا المثال بالملف output.dat ، وللتأكد من المحتوى استعمل الأمر TYPE الذي يعرض الآتي :-

```
Here is file [ output.dat ]
***********************
   1) Number of blanks ==> 12
   2) Number of letters ==> 69
   3) Number of digits ==> 6
   4) Number of lines ==> 4
```

وكما نلاحظ فإن هذه الإحصائية مطابقة للبيانات الموجودة في الملف input.dat

4.14) الملفات الثنائية Binary Files

إضافة إلى الملفات النصية توجد الملفات الثنائية binary files تدون فيها الرموز الموجودة في لوحة المفاتيح سواء القابلة للطباعة مثل الأرقام والحروف الهجائية أو غير القابلة للطباعة مثل المفاتيح Ins, Del, Home, End وغيرها في صورتها الأصلية أي شفرة (ascii) .

1) الكتابة في الملف Writing in a File

مثال (14-4-1)

البرنامج التالي يوضح حفظ البيانات في ملف ثنائي .

```
#include <stdio.h>
#include <conio.h>
#define filename "PROG2.DAT"
main()
{
    struct
    {
      char str[20];
      int num;
    } both;
    FILE *fpt;
    clrscr();
    fpt = fopen(filename, "wb");
    printf("Type your string ==> ");
    gets(both.str);
    printf("Type your integer ==> ");
    scanf("%d", &both.num);
    fwrite(&both, sizeof(both), 2, fpt);
    fclose(fpt);
}
```

في هذا البرنامج خصص اسم الملف PROG2.DAT للثابت الرمزى filename عن طريق المعالج الأولي define ، والإعلان عن التركيبة both

وتضم عنصرين الأول نوعه حرفي والثاني نوعه صحيح ، بعد فتح الملف الثنائي prog2.dat وإدخال قيمة المتغير الأول both.str والمتغير الثاني both.num، جاءت الدالة ()fwrite وشكلها :

```
fwrite(position , size , max , pointer_name);
```

حيث :-

position	تعني مؤشر يشير إلى موقع التخزين ، حيث تم استخدام الرمز (&) قبل التركيبة both .
size	يعني الحجم أو الحيز الذي تشغله البيانات المدخلة بالبايت ، فمثلا : sizeof(both) هنا المؤثر sizeof مهمته حساب السعة في ذاكرة الحاسب للمتغيرين num, str
max	تعني عدد الحقول (Fields) المدخلة .
pointer_name	مؤشر الملف المطلوب كتابة قيم المتغيرين num, str فيه وهو PROG2.DAT

وقت تنفيذ هذا البرنامج ، تكون المدخلات كما يلي :-

```
Type your string ==> Welcome user
Type your integer ==> 1234
```

2) قراءة الملف Reading a File

مثال (14-4-2)

البرنامج الآتي مهمته قراءة البيانات التي تم حفظها في الملف prog2.dat عن طريق البرنامج بالمثال (14-4-1) مع إظهارها على شاشة العرض .

```
#include <stdio.h>
#include <conio.h>
```

```
#include <process.h>
main()
{
    struct
    {
      char string[20];
      int num;
    } both;
    FILE *fpt;
    clrscr();
    fpt = fopen("PROG2.DAT" , "rb");
    if( fpt == NULL )
    {
      printf("CANNOT OPEN FILE");
      exit(0);
    }
    clrscr();
    fread(&both, sizeof(both), 2, fpt);
    printf("Your integer ==> %d ", both.num);
    printf("\nWhile your string ==> %s ", both.string);
    fclose(fpt);
}
```

هنا تم استخدام الدالة fread() التي لها نفس شكل الدالة fwrite() وذلك لقراءة البيانات المحفوظة بالملف PROG2.DAT وطباعتها بالصورة التالية :

```
Your integer ==> 1234
While your string ==> Welcome user
```

مثال (14-4-3)

المطلوب كتابة برنامج يقوم بمعالجة البيانات المدخلة خلال اليوم في مصرف ما ، حيث يتم قراءة رقم الحساب وعدد الصكوك وقيمة الصكوك ثم تخزين هذه المعلومات في ملف ثنائي .

```
#include <stdio.h>
#include <conio.h>
#define NUM   5
main()
{
    FILE *bank;
    int I, checks;
    long acctno;
    float amount, total = 0.0;
    clrscr();
    bank = fopen("BANK.DAT" , "wb+");
    for(i = 0 ; i < NUM ; i++)
    {
      printf("\nEnter account number ");
      printf(" and number of checks : ");
      scanf("%ld %d", &acctno, &checks);
      printf("Enter amount of checks :");
      scanf("%f", &amount);
      fwrite(&acctno, sizeof(long), 1, bank);
      fwrite(&checks, sizeof checks, 1, bank);
      fwrite(&amount, sizeof(float), 1, bank);
    }
    rewind(bank);
    printf("\naccount no.  number of checks  amount\n");
    for(i = 0 ; i < NUM ; I++)
    {
      fread(&acctno, sizeof(long), 1, bank);
      fread(&checks, sizeof(int), 1, bank);
      fread(&amount, sizeof(float), 1, bank);
      total += amount;
      printf(" %ld    %d    %.2f\n", acctno, checks, amount);
    }
    printf("Total amount of checks ==> %.2f", total);
    fclose(bank);
}
```

بعد تحديد عدد الزبائن M الذين تعاملوا مع المصرف ، كالمعتاد تم إشهار مؤشر الملف bank ومن ثم جرى إنشاء وفتح ملف تحت اسم BANK.D على أساس أنه ملف ثنائي ، حيث استخدمت حروف التعامل (wb+) التي تعني أن الملف قابل للكتابة فيه والقراءة منه ، وتم إدخال رقم الحساب acctno وعدد

الصكوك checks وقيمة هذه الصكوك amount ، وبالتالي كتابة رقـم الحسـاب عن طريق الدالة fwrite() بالشكل الآتي :

```
fwrite(&acctnc; sizeof(long), 1, bank);
```

حيـث اسـتخدم الرمـز (&) قبـل المتغير acctno ليـدل علـى تخزيـن رقـم الحساب الذي هو من النوع الطويل long أما المؤثر sizeof فيعني حساب عدد الخانات للمتغير acctno حيث وضع النوع long بين القوسين ، أما إذا اسـتخدم هذا المؤثر مع المتغير مباشـرة فـلا داعـي لاسـتخدام الأقـواس كمـا فـي دالـة fwrite() التالية ، يلي هذا المؤثر الرقـم 1 الـذي يمثـل عـدد البيانـات المدخلـة، أخيراً جاء مؤثر الملف وهو bank .

بعد الانتهاء من إدخال البيانات لعدد 5 زبائن ، التي قد تكون كالآتي :-

```
Enter account number and number of checks : 3456  3
Enter amount of checks : 123.50

Enter account number and number of checks : 1230  9
Enter amount of checks : 760.39

Enter account number and number of checks : 7620  6
Enter amount of checks : 664.4

Enter account number and number of checks : 5391  8
Enter amount of checks : 865.05

Enter account number and number of checks : 3574  2
Enter amount of checks : 904.64
```

يتم إرجاع مؤشر الملف الثنائي bank إلى بداية الملف عن طريـق الدالـة rewind() وبالتالي تنفيذ جملة while بقصد قراءة المعلومات عن طريق الدالـة fread() مع إيجاد المجمـوع الكلـي للصكـوك المدخلـة وطباعتهـا علـى شاشـة العرض فى شكل يشبه الآتـي :-

```
account no.  number of checks  amount
3456            3              123.50
1230            9              760.39
7620            6              664.40
5391            8              865.05
3574            2              904.64
Total amount of checks ==>  3317.98
```

5.14) الوصول إلى البيانات Access to Data

فى أحيان كثيرة نحتاج إلى معالجة البيانات المخزنة بالملفات ، ولكن قبل الشروع فى معالجتها لا بد أن نتعرف على الطرق الموصلة إليها ، ولكى نصل إليها هناك طريقتان هما :

1) الوصول التتابعي Sequential Access

حيث يتحتم المرور على كل السجلات قبل الوصول إلى السجل المطلوب معالجته ، وعند إضافة أي سجل جديد فسوف تتم كتابته في آخر الملف والسبب أن السجلات التى يتكون منها الملف مرتبة ترتيبا تصاعديا حسب التخزين .

2) الوصول المباشر Direct Access

وهي طريقة أكثر ملاءمة لمعالجة البيانات بالملفات حيث تحتاج إلى وقت أقل من سابقتها ونستطيع بواسطتها إضافة أو إلغاء أي سجل وفي أي موقع من الملف بسرعة كبيرة .

مثال (14-5-1)

الغرض من البرنامج الآتي معالجة البيانات المحفوظة في الملف تحت اسم STORE.DAT وهي رقم الصنف وثمنه خاصة بمخزن يحتوي على عدد من الأصناف ، والتعديل في أي خانة من الخانات التى يحتوى عليها السجل المخزن بالملف .

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
main()
{
    struct
    {
      long itemno;
      double price;
    } both;
    char op, response, filename[20];
    long postion;
    FILE *store;
    clrscr();
    printf(" Type your file name ==> ");
    gets(filename);
    store = fopen(filename, "rb+");
    if( store == NULL )
    {
       printf(" Error file [ %s ] not found", filename);
       exit(0);
     }
     do
     {
      printf("\n Enter item number ==> ");
      scanf("%ld", &both.itemno);
      postion = (both.itemno) * sizeof(both);
      fseek(store, postion, SEEK_SET);
      fread(&both.price, sizeof(double), 1, store);
      printf("\n Old record as following :-");
      printf("\n\t Item number is %ld  ", both.itemno);
      printf("and price is %.3lf", both.price);
      printf("\n Do you like to change ");
      printf("the price [Y/N] ? : ");
      getchar();
      scanf("%c", &op);
      if( (op == 'Y') || (op == 'y') )
      {
        printf("\n Enter new price ==> ");
        scanf("%lf", &both.price);
        fseek(store, postion, SEEK_SET);
        fwrite(&both.price, sizeof(double), 1, store);
      }
```

```
        printf("\n Do you like to continue [Y/N] ? ");
        getchar();
        scanf("%c", &response);
      }
      while( response == 'Y' || response == 'y' );
      fclose(store);
}
```

عند تنفيذ البرنامج يحصل نوع من التحاور بين البرنامج ومستعمله كالآتي:-

```
Type your file name ==> store.dat
Enter item number ==> 2

Old record as following :-
     Item number is 2 and price is 2750.990

Do you like to change the price [Y/N] ? : Y

Enter new price ==> 750.99
Do you like to continue [Y/N] ? Y

Enter item number ==> 1
Old record as following :-
     Item number is 1 and price is 537.760

Do you like to change the price [Y/N] ? : Y

Enter new price ==> 5037.76
Do you like to continue [Y/N] ? Y

Enter item number ==> 2
Old record as following :-
     Item number is 2 and price is 750.990

Do you like to change the price [Y/N] ? : N

Do you like to continue [Y/N] ? N
```

يستحسـن عنـد معالجـة الملفـات أن يكـون شـكل عناصرهـا علـى هيئـة تركيبة (Structure) حتي يسهل معالجة عناصر أو خانات هذه التركيبة ، وبهذا تم استخدام التركيبة both التي تحتوي على عنصريـن ، رقـم الصنـف itemno من النوع الطويل وثمنه price من النوع المضاعف .

بعـد تخزيـن الملـف STORE.DAT ننتقل لمعالجـة بياناتـه ولتحقيـق ذلـك ، وجب عمل الآتي :-

1) للوصول إلى أي سجل في الملف، يتم ذلك عن طريق المفتاح key وهو رقم الصنف itemno في هذه الحالة حيث تم إدخال الرقم 265 مثلاً.
2) تحديد حجم أو طول السجل في الملـف المطلوب معالجته عن طريق المؤثر sizeof والجملة :

```
postion = (both.itemno) * sizeof(both);
```

التي مهمتها تحديد قيمـة المؤشـر الـذي يشـير إلـى مكـان رقـم الصنـف both.itemno مضروبا في طول هذا السجل بالبايت عن طريـق المؤثـر sizeof .

3) استخدمت الدالة fseek() وشكلها :

```
fseek(file_pointer , offset , place);
```

وتعني تحديد مكان عملية الإدخال أو الإخراج التالية حيث :-

offset متغير من النوع الطويل long .
place متغير من النوع الصحيح int .

أي تعييـن بدايـة القـراءة أو الكتابـة مـن الملـف المشـار إليـه بواسطة file_pointer الذى هو تحت اسم STORE.DAT .

يمكن أن يأخذ المتغير place احدى القيم الثلاث التالية :-

1) القيمة 0 أو SEEK_SET وتعني من بداية الملف .

2) القيمة 1 أو SEEK_CUR وتعني قياس موضع القراءة أو الكتابة من الموقع الحالي للملف .

3) القيمة 2 أو SEEK_END وتعني من نهاية الملف .

مع مراعاة أن تكتب هذه القيم المعرفة بالحروف الكبيرة ، فالأمر :-

```
fseek(store, postion, SEEK_SET);
```

يعني البحث عن السجل بعنوان postion من بداية الملف store لغرض قراءته وبالتالي إمكانية التعديل فيه .

تم استخدام الدالة fread() لقراءة ثمن الصنف الذي رقمه 2 وبالتالي طباعة كل البيانات الخاصة بهذا السجل على شاشة العرض .

بعدها يتم الاختيار بين التعديل في السجل من عدمه ، فإذا كان الاختيار بنعم (Y) عندها يطلب الحاسب إدخال الثمن الجديد والبحث عنه وتحديد موقعه ثم الكتابة في ذلك الموقع عن طريق الدالة fwrite() حيث تم تعديل ثمن الصنف رقم 2 من 2750.99 دينار إلى 750.99 ديناراً ، وكذلك الصنف رقم 1 من 537.76 ديناراً إلى 5037.76 ، وتستمر معالجة هذا الملف حتى إدخال الحرف (N) عندها يتم إيقاف تنفيذ البرنامج .

آخر الأمثلة: منظومة مكتبة

المطلوب إعداد برنامج مهمته تصميم شاشة العرض بحيث تحتوي على عدد من الاختيارات تضم:-

1) إضافة كتب جديدة.

2) إلغاء كتب.

3) حصر الكتب المستعارة.

4) حصر الكتب المفقودة.

5) عرض الكتب في كل حالة ذكرت اعلاه.

الحل:

تم تقسيم هذه المسألة إلى عدد من الدوال الفرعية بدلاً من كتابتها معاً في برنامج واحد حتى يسهل تنفيذ ومتابعة وفهم كل دالة على حدة وهي:-

الدالة الأولى window_1() التي بدورها تحتوي على :-

1) دالة تلوين أرضية شاشة العرض textbackground() حيث يستخدم رقم أو متغير من النوع الصحيح بين القوسين () ومداه من 0 إلى 15 ، علما بأنه يوجد في لغة C عدد من الألوان يمكن الاستفادة منها في كثير من البرامج حتى تعطي خلفية جميلة للمستخدم ، انظر الملحق (2) .

2) دالة تكوين نافذة فرعية حيث يتم تخصيص مساحة من شاشة العرض كنافذة صغيرة تأخذ الشكل الآتي:-

```
window(X1, Y1, X2,Y2);
```

حيث Y1, X1 هما إحداثيات الجزء العلوي ناحية اليسار من النافذة .

و Y2, X2هما إحداثيات الجزء السفلي ناحية اليمين من النافذة .

خذ مثلا الأمر :

```
window(1, 1, 80, 25);
```

يعني تكوين نافذة مستطيلة أبعادها بداية من العمود 1 والسطر 1 بطول 80 عمودا من الناحية اليمني وبعرض 25 سطرا إلى اسفل الشاشة .

وبالطبع يجب الأخذ في الاعتبار أن مدى شاشة العرض هو 80 عمودا في 25 سطرا .

3) تنظيف النافذة التي أعدت سابقا عن طريق الدالة clrscr() .

4) الدالة ()textcolor وهي تختص بتلوين الحروف المراد إظهارها على النافذة .

الدالة الثانية ()window_2 مهمتها إعطاء اللون الأزرق بكتابة الرقم 1 بين الأقواس عن طريق الدالة (1)textbackground لأرضية النافذة الرئيسية ;(window(17, 3, 66, 23 حيث تمت كتابة الجملة SCREEN SHEET داخل هذه النافذة إبتداء من العمود 12 والسطر 2 باللون الاحمر .

بعدها قسمت النافذة الرئيسية إلى خمس نوافذ فرعية أرضيتها باللون الأبيض ومكتوب فيها الحروف باللون الأحمر وهذه النوافذ تم تصميمها عن طريق :

الدالة الثالثة ()window_3 التى تختص بإظهار الرسالة :

**********> [A] ADD NEW BOOKS <*******

لإدخال الحرف A بقصد إضافة كتب جديدة .

الدالة الرابعة ()window_4 التى تختص بإظهار الرسالة :

**********> [D] DELETE BOOK <***********

إدخال الحرف D لإلغاء أي كتاب .

الدالة الخامسة ()window_5 التى مهمتها عرض الرسالة :

**********> [S] SHOW ALL BOOKS <*********

أي أظهار كل محتويات المكتبة من كتب عن طريق إدخال الحرف S .

الدالة السادسة ()window_6 التى مهمتها إظهار EXIT < ESC >

أي الضغط على مفتاح Esc للخروج .

وهي نافذة الاختيارات حيث تظهر الرسالة

الدالة السابعة ()window_7 وهي نافذة الاختيارات حيث تظهر الرسالة :

CHOICE < A-D-S >

يلي ذلك تنفيذ النافذة المقابلة لأي حرف من هذه الحروف .

وفيما يلي البرنامج كاملا

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
#define ESC  27
FILE *fpt, *fpt1 , *fpt2 , *fpt3;
int day,month,year,book_no,status,book_id;
float price, sum;
char book_title[20], author_name[20], dept[20],ch;
main()
{
    void window_1(); void window_2();
    void window_3(); void window_4();
    void window_5(); void window_6();
    void window_7(); void add_rec();
    void delete_rec(); void print_all();
    for( ; ; )
    {
      window_1();
      window_2();
      window_3();
      window_4();
      window_5();
      window_6();
      window_7();
      ch = getch();
      switch(ch)
      {
        case 'a' :
        case 'A' : add_rec(); break;
        case 'd' :
        case 'D' : delete_rec(); break;
        case 's' :
        case 'S' : print_all(); break;
        case ESC  : exit(0);
      }
    }
}
```

```
void window_1()
{
    textbackground(7);
    window(1, 1, 80, 25);
    clrscr();
    textcolor(4);
}

void window_2()
{
    textbackground(1);
    window(17, 3, 66, 23);
    clrscr();
    gotoxy(12, 2);
    printf("<   SCREEN   SHEET   >");
}

void window_3()
{
    textbackground(7);
    window(19, 6, 64, 8);
    clrscr();
    gotoxy(2, 2);
    printf("**********> [ A ] ADD NEW BOOKS <*******");
}

void window_4()
{
    textbackground(7);
    window(19, 10, 64, 12);
    clrscr();
    gotoxy(2, 2);
    printf("**********> [ D ] DELETE BOOK <***********");
}

void window_5()
{
    textbackground(7);
    window(19, 14, 64, 16);
    clrscr();
    gotoxy(2, 2);
    printf("**********> [ S ] SHOW ALL BOOKS <*********");
```

```
}

void window_6()
{
    textbackground(7);
    window(45, 20, 60, 22);
    clrscr();
    gotoxy(4, 2);
    printf("EXIT < ESC > ");
}

void window_7()
{
    textbackground(7);
    window(23, 18, 41, 22);
    clrscr();
    gotoxy(2, 2);
    printf("CHOICE < A-D-S >\n");
    gotoxy(3, 4);
    printf("[  ]");
}

/* ---------------- ADD FUNCTION --------------- */
void add_rec()
{
   int n,i;
   window_1();
   printf(" ENTER  NUMBER OF BOOKS TO ADD : ");
   scanf("%d", &n);
   fpt = fopen("LIBRARY.DAT" , "a");
   printf("ENTER DATA AS :- \n   STATUS  BOOK NO  BOOK NAME");
   printf("AUTHOR NAME   BOOK PRICE  DAY/MONTH/YEAR \n");
    for(i = 1 ; i <= n ; ++i)
    {
          scanf("%d%d%s      %s",&status,      &book_no,      book_title,
author_name);
       scanf("%s%f%d%d%d", &dept, &price, &day, &month, &year);
       fprintf(fpt,"%d  %d    %s %s    %s    %6.2f  %2d %2d %2d\n",
               status, book_no, book_title, author_name, dept,
               price, day, month, year);
    }
    fclose(fpt);
 }
```

```
/* ----------------- DELETE FUNCTION --------------- */
void delete_rec()
{
    window_1();
    printf("ENTER BOOK_NO TO DELETE : ");
    scanf("%d", &book_id);
    fpt = fopen("LIBRARY.DAT" , "r");
    fpt3 = fopen("LIBRARY.OUT" , "w");
    while( !feof(fpt) )
    {
      fscanf(fpt,"%d%d%s%s%s%f%d%d%d\n",
              &status, &book_no, &book_title, &author_name,
              &dept, &price, &day, &month, &year);
      if( book_no != book_id )
        fprintf(fpt3,"%d %d %s %s %s %f %d %d %d\n",
                status, book_no, book_title, author_name,
                dept, price, day, month, year);
    }
    fprintf(fpt3,"\f");
    fclose(fpt3); fclose(fpt);
    fpt3 = fopen("LIBRARY.OUT" , "r");
    fpt = fopen("LIBRARY.DAT" , "w");
    while( !feof(fpt3) )
    {
      fscanf(fpt3,"%d%d%s%s%s%f%d%d%d\n",
              &status, &book_no, &book_title, &author_name,
              &dept, &price, &day, &month, &year);
      if( status >= 0 )
        fprintf(fpt,"%d %d %s %s %s %f %d %d %d\n",
                status, book_no, book_title, author_name,
                dept, price, day, month, year);
    }
    fclose(fpt3); fclose(fpt);
}

/* ---------------- PRINT FUNCTION ---------------- */
void print_all()
{
    int lost_books(int , int , char name[20], float );
    int borrow_books(int , char dept[20], int , int, int );
    sum = 0.0;
    window_1();
    fpt = fopen("LIBRARY.DAT" , "r");
```

```
printf("\n LIST OF ALL BOOKS ");
printf("\n ***************\n\n");
printf("\nSTATUS  NUMBER  NAME    AUTHOR NAME");
printf("  DEPT   COST   DATE\n\n ");

while ( !feof(fpt) )
{
  fscanf(fpt,"%d%d%s%s%s%f%d%d%d\n",
          &status, &book_no, &book_title, &author_name,
          &dept, &price, &day, &month, &year);
  printf("\n%d %10d    %s    %s    %s  %.2f  %d/%d/%d",
           status, book_no, book_title, author_name, dept,
           price, day, month, year);
}
 printf("\n\n PRESS ANY KEY TO CONTINUE ");
 getch();
 fpt = fopen("LIBRARY.DAT" , "r");
 fpt1 = fopen("LOSTBOOK.DAT" , "w");
 fprintf(fpt1,"LOST BOOK \n");
 fprintf(fpt1,"**********\n\n");
 fprintf(fpt1,"BOOK_NUMBER   BOOK_NAME  BOOK_PRICE\n");
 fprintf(fpt1,"----------------------------------\n");
 fpt2 = fopen("BROWBOOK.DAT" , "w");
 fprintf(fpt2,"BORROWED ALL BOOK \n");
 fprintf(fpt2,"**************** \n");
 fprintf(fpt2," BOOK_NUMBER DEPARTMENT_NAME   DATE\n");
 fprintf(fpt2," ------------------------------------------\n");
 while( !feof(fpt) )
 {
  fscanf(fpt,"%d%d", &status, &book_no);
  fscanf(fpt,"%s", &book_title);
  fscanf(fpt,"%s", &author_name);
  fscanf(fpt,"%s%f", &dept,&price);
  fscanf(fpt,"%d%d%d\n", &day, &month, &year);
  if( status != 0 )
  {
    if( status == 1 )
         sum += price;
    lost_books(status, book_no, book_title, price);
  }
     else
         borrow_books(book_no, dept, day, month, year);
 }
```

```
        fprintf(fpt1,"\t\t\tSUM = %.2f", sum);
        fclose(fpt);  fclose(fpt1);    fclose(fpt2);
}

int lost_books(int status, int num, char name[20], float cost)
{
    switch(status)
    {
      case 1 : fprintf(fpt1,"%d%20s", num, name);
               fprintf(fpt1,"    %8.2f\n", cost);
               break;
      case 2 : break;
    }
}

int borrow_books(int num, char dept[20], int dd, int mm, int yy)
{
    fprintf(fpt2," %d %22s", num, dept);
    fprintf(fpt2," %11d/%d/%d\n", dd, mm, yy);
}
```

فمثلا عندما يريد المستعمل إضافة كتب جديدة إلى ملف البيانات الرئيسي ، فما عليه إلا إدخال الحرف A أو الحرف a ، عندها ينتقل التحكم عن طريق جملة switch إلى استدعاء الدالة add_rec() لإضافة كتب جديدة ويظهر السطر :

ENTER NUMBER OF BOOKS TO ADD :

على الشاشة ، عندها يرد المستعمل بإدخال عدد الكتب المراد إضافتها ، ويتم البحث عن الملف LIBRARY.DAT فتحه الذى من المفترض أن يكون موجودا قبل تنفيذ هذا البرنامج ويحتوي على البيانات الرئيسية للمكتبة بالترتيب التالي :-

1) حالة الكتاب (status) من النوع الصحيح ، 0 يعني أن الكتاب مستعار، 1 الكتاب مفقود ، وأية قيمة اخرى تعني أن الكتاب موجود بالمكتبة .
2) رقم الكتاب (book_no) من النوع الصحيح .
3) عنوان الكتاب (book_title) من نوع السلسلة الحرفية .
4) اسم المؤلف (author_name) من نوع السلسلة الحرفية .

5) القسم التابع له الكتاب (dept) من نوع السلسلة الحرفية .
6) ثمن الكتاب (price) من النوع الحقيقي .
7) تاريخ الاستعارة اليوم (day) ، الشهر (month) ، السنة (year) .

تطلب هذه الدالة عدد الكتب المراد إضافتها إلى ملف البيانات الرئيسي ، وإدخالها بالترتيب المبين اعلاه .

ترجع النافذه الرئيسية مرة ثانية وفي حالة إدخال الحرف D ينتقل التحكم إلى استدعاء الدالة delete_rec() ويظهر السطر

ENTER BOOK_NO TO DELETE :

أي إلغاء كتاب معين من الملف LIBRARY.DAT حيث يتم إدخال رقم الكتاب المطلوب إلغاؤه من الملف عن طريق المتغير book_id ، حيث يبدأ البحث عنه بواسطة جملة

```
while( !feof (fpt) )
```

التي عن طريقها يتم قراءة كل سجل من السجلات مع مقارنة الرقم المدخل book_id مع رقم الكتاب book_no فإذا ما وجد يتم إلغاؤه مع حفظ البيانات المعدلة في ملف البيانات الرئيسي LIBRARY.DAT .

بإدخال الحرف S من خلال اختيار نافذة الطباعة ، حينها تستدعى print_all() ومهمتها :

1) عرض محتويات الملف الرئيسي القديم أو المعدل LIBRARY.DAT باللون الأحمر على شاشة بيضاء .
2) استدعاء الدالة lost_bcoks() ، والغرض منها حساب عدد الكتب المفقودة ومن ثم حفظها مع مجموع أسعارها في ملف تحت اسم LOSTBOOK.DAT.
3) استدعاء الدالة borrow_book() لحفظ الكتب المستعارة مع تاريخ استعارتها في ملف آخر تحت اسم BOROWBOOK.DAT .

فيما يلي قائمة بالاختيارات التي يحتويها هذا البرنامج عند التنفيذ :

SCREEN SHEET

```
**********> [ A ] ADD NEW BOOKS <*******
**********> [ D ] DELETE BOOK <***********
**********> [ S ] SHOW ALL BOOKS <*********
CHOSE < A-D-S >          EXIT < ESC >
```

أما بخصوص الملفات التي يضمها هذا البرنامج فهي :-

1) الملف الرئيسي LIBRARY.DAT الذي يضم قائمة بجميع الكتب بالمكتبة وقد يكون بالشكل الآتي :-

```
3  16 BOTANY      MOHAMMED  BOTANY     7.5   3  8  1999
1  14 C++         GAYED     COMPUTER   14.5  0  0  0
0  17 BOBOL       WINCHUNG  COMPUTER   11.5  5  7  1999
1  21 FORTRAN     JALAL     COMPUTER   12.5  0  0  0
0  11 FORTRAN77   LOREN     COMPUTER   7.0   1  5  1999
1  12 PASCAL      GAYED     COMPUTER   12.0  0  0  0
2  27 PHYSICS     AHMED     PHYSICS    9.7   0  0  0
0  13 STATICS     SALEM     STATICS    8.5   5  2  1998
```

2) الملف الثاني BROWBOOK.DAT وهو يضم الكتب المستعارة وحالتها (0) وهي :-

```
BORROWED ALL BOOK
*************************
BOOK_NUMBER   DEPARTMENT_NAME     DATE
-------------------------------------------------------
    17              COMPUTER          5/7/1999
    11              COMPUTER          1/5/1999
    13              STATICS           5/2/1998
```

3) الملف الثالث LOSTBOOK.DAT ويضم قائمة بجميع الكتب المفقودة وحالتها (1) مع مجموع أسعارها وهي :-

```
LOST BOOK
******************
BOOK_NUMBER   BOOK_NAME       BOOK_PRICE
-------------------------------------------------
    14          C++             14.50
    21          FORTRAN         12.50
    12          PASCAL          12.00

              SUM =       39.00
```

6.14) تمرينات Exercises

1) اذكر الفرق بين :

a) Sequential Files & Direct Files
b) Binary Files & Text Files
c) File & Record & Field

2) أعد كتابة التمرين (4) بالفصل السابق بحيث يتم تخزين كل البيانات في ملف تحت اسم pharmacy ، وأيضاً إنشاء ملف آخر يضم أسماء الأدوية وجهة إنتاجها وأسعارها والمنتهية صلاحيتها (حدد تاريخ الانتهاء) .

3) اكتب برنامجا لقراءة وتخزين البيانات التالية في ملف ثنائي Binary :

* رقم الزبون Customer Number .
* العنوان Address .
* تاريخ آخر المشتروات Date of last Purchase وتضم الشهر Month والسنة Year .
* قيمة الدين Amount Owed .

ثم القيام بتعديل هذه البيانات باستخدام رقم الزبون وتخزينها في ملف آخر يحتوي على رقم الزبون وتاريخ آخر المشتروات مع قيمة الدين .

4) اكتب برنامجا يستقبل عددا من السطور عن طريق لوحة المفاتيح ويخزنها في ملف نصي Text File ثم اكتب برنامجا آخر مهمته قراءة البيانات الموجودة في الملف السابق وطباعة محتويات هذا الملف وعدد سطوره وعدد الرموز التي يحتويها وعدد الكلمات التى يتكون منها كل سطر .

5) على فرض أن لدينا ملفين الأول يحتوي على رقم الموظف Employee number واسمه Name ونوع العمل Job Type .

والثاني به رقم الموظف Employee number وراتبه Salary وتاريخ تعيينه Date .

المطلوب قراءة البيانات بالملفين السابقين مع استحداث ملف جديد تخزن به البيانات بالشكل الآتي :-

Employee Number	Job name	Salary
...	...	...
...	...	...
...	...	...

Total Salary = ...

المطلوب أيضاً انشاء ملف آخر يضم قائمة بأرقام الموظفين ونوع عملهم مع تاريخ تعيينهم .

6) المطلوب كتابة برنامج لقراءة البيانات من ملف البيانات الذى أعد في التمرين (4) بالفصل السابق مع تنفيذ التالي :-

- طباعة تقرير كامل بمحتويات الصيدلية من الدواء .
- التعديل في ثمن الدواء عن طريق رقم الدواء .
- طباعة تقرير بكل الأدوية التي انتهت صلاحيتها للجهة المنتجة ، مع حذفها من الملف الرئيسي .
- إضافة دواء جديد .

7) اكتب برنامجا ، يتم فيه ادخال رقم واسم عدد من الطلبة مع المبلغ المطلوب من الطالب مقابل تسجيله ، وإخراج تقرير يضم الطلبة الذين لم يقوموا بدفع اشتراكاتهم في الملف الأول ، وأرقام الطلبة والمواد التي تم تسجيلها في ملف آخر .

8) اكتب برنامجا كاملا ينتج عنه استدعاء دالة مهمتها قراءة بيانات تخص مصرف الدم وتخزينها في ملف رئيسي والبيانات هي : رقم الشخص ، اسمه ، عنوانه ، رقم الهاتف ، العمر ، الفصيلة .

مع عمل ما يأتي :-

* طباعة رقم الشخص ، الاسم ، رقم الهاتف للأشخاص الذين لديهم فصيلة دم A موجبة .
* تخزين الاسم ، العنوان ، العمر للأشخاص الذين لديهم فصيلة الدم O سالبة أو A سالبة ، مع عددهم في ملف أخر .
* انشاء ملف ثالث تخزن فيه كل التعديلات وأي إضافة أو إلغاء للبيانات الموجودة بالملف الرئيسي.
* إنتاج قائمة بكل الأسماء وارقام هواتف الذين تكون أعمارهم أقل من أو تساوي 18 سنة ولديهم فصيلة الدم B سالبة أو موجبة.

9) يحتفظ أحد المصارف بملف يضم بيانات تخص كل الزبائن وهي رقم الحساب ، اسم الزبون ، الرصيد الحالى ، عنوان الزبون ، تاريخ آخر تعديل ، نوع المعاملة وهي 1 حساب جديد ، 2 إيداع ، 3 سحب ، 4 قفل حساب . المطلوب كتابة برنامج لإنشاء :

* ملف يضم رقم الحساب والعنوان لكل زبون أقفل حسابه .
* ملف يضم رقم الحساب والمبالغ المودعة والمسحوبة .
* ملف يضم كل التعديلات وهي رقم الحساب وتاريخ المعاملة والمبلغ مع نوع المعاملة التي تمت .

الملاحــق

ملحق (1): جدول أولويات تنفيذ العمليات

المؤثر	التنفيذ
() [] -> .	من اليسار إلى اليمين
! ++ -- -(unary)	من اليمين إلى اليسار
* / %	من اليسار إلى اليمين
+ -	من اليسار إلى اليمين
<< >>	من اليسار إلى اليمين
< <= > >=	من اليسار إلى اليمين
== !=	من اليسار إلى اليمين
&	من اليسار إلى اليمين
^	من اليسار إلى اليمين
|	من اليسار إلى اليمين
&&	من اليسار إلى اليمين
||	من اليسار إلى اليمين
? :	من اليمين إلى اليسار
= += -= *= /= %= <<= >>=	من اليمين إلى ايسار

ملحق (2): جدول الألوان

الرمز	اللون	الرقم	اللون
0	أسود	8	رمادي غامق
1	أزرق	9	أزرق فاتح
2	أخضر	10	أخضر فاتح
3	كحلي	11	كحلي فاتح
4	أحمر	12	أحمر فاتح
5	أرجواني	13	أرجواني فاتح
6	بني	14	أصفر
7	رمادي داكن	15	أبيض

ملحق (3)

جدول رموز آسكي ASCII

القيمة بالنظام									
عشري	ثنائي	ثماني	ستة عشري	ascii	عشري	ثنائي	ثماني	ستة عشري	ascii
0	00000000	000	0x0	NUL	65	01000001	101	0x41	A
1	00000001	001	0x1	SOH	66	01000010	102	0x42	B
2	00000010	002	0x2	STX	67	01000011	103	0x43	C
3	00000011	003	0x3	ETX	68	01000100	104	0x44	D
4	00000100	004	0x4	EOT	69	01000101	105	0x45	E
5	00000101	005	0x5	ENQ	70	01000110	106	0x46	F
6	00000110	006	0x6	ACK	71	01000111	107	0x47	G
7	00000111	007	0x7	BEL	72	01001000	110	0x48	H
8	00001000	010	0x8	BS	73	01001001	111	0x49	I
9	00001001	011	0x9	HT	74	01001010	112	0x4A	J
10	00001010	012	0xA	LF	75	01001011	113	0x4B	K
11	00001011	013	0xB	VT	76	01001100	114	0x4C	L
12	00001100	014	0xC	FF	77	01001101	115	0x4D	M
13	00001101	015	0xD	CR	78	01001110	116	0x4E	N
14	00001110	016	0xE	SO	79	01001111	117	0x4F	O
15	00001111	017	0xF	SI	80	01010000	120	0x50	P
16	00010000	020	0x10	DLE	81	01010001	121	0x51	Q
17	00010001	021	0x11	DC1	82	01010010	122	0x52	R
18	00010010	022	0x12	DC2	83	01010011	123	0x53	S
19	00010011	023	0x13	DC3	84	01010100	124	0x54	T
20	00010100	024	0x14	DC4	85	01010101	125	0x55	U

القيمـــــة بالنظـــــام									
عشري	ثنائي	ثماني	ستة عشري	ascii	عشري	ثنائي	ثماني	ستة عشري	ascii
21	00010101	025	0x15	NAK	86	01010110	126	0x56	V
22	00010110	026	0x16	SYN	87	01010111	127	0x57	W
23	00010111	027	0x17	ETB	88	01011000	130	0x58	X
24	00011000	030	0x18	CAN	89	01011001	131	0x59	Y
25	00011001	031	0x19	EM	90	01011010	132	0x5A	Z
26	00011010	032	0x1A	SUB	91	01011011	133	0x5B	[
27	00011011	033	0x1B	ESC	92	01011100	134	0x5C	\
28	00011100	034	0x1C	FS	93	01011101	135	0x5D	]
29	00011101	035	0x1D	GS	94	01011110	136	0x5E	^
30	00011110	036	0x1E	RS	95	01011111	137	0x5F	_
31	00011111	037	0x1F	US	96	01100000	140	0x60	`
32	00100000	040	0x20	SP	97	01100001	141	0x61	a
33	00100001	041	0x21	!	98	01100010	142	0x62	b
34	00100010	042	0x22	"	99	01100011	143	0x63	c
35	00100011	043	0x23	#	100	01100100	144	0x64	d
36	00100100	044	0x24	$	101	01100101	145	0x65	e
37	00100101	045	0x25	%	102	01100110	146	0x66	f
38	00100110	046	0x26	&	103	01100111	147	0x67	g
39	00100111	047	0x27	'	104	01101000	150	0x68	h
40	00101000	050	0x28	(	105	01101001	151	0x69	I
41	00101001	051	0x29	)	106	01101010	152	0x6A	j
42	00101010	052	0x2A	*	107	01101011	153	0x6B	k
43	00101011	053	0x2B	+	108	01101100	154	0x6C	l

القيمـــــة بالنظـــــام									
عشري	ثنائي	ثماني	ستة عشري	ascii	عشري	ثنائي	ثماني	ستة عشري	ascii
44	00101100	054	0x2C	,	109	01101101	155	0x6D	m
45	00101101	055	0x2D	-	110	01101110	156	0x6E	n
46	00101110	056	0x2E	.	111	01101111	157	0x6F	o
47	00101111	057	0x2F	/	112	01110000	160	0x70	p
48	00110000	060	0x30	0	113	01110001	161	0x71	q
49	00110001	061	0x31	1	114	01110010	162	0x72	r
50	00110010	062	0x32	2	115	01110011	163	0x73	s
51	00110011	063	0x33	3	116	01110100	164	0x74	t
52	00110100	064	0x34	4	117	01110101	165	0x75	u
53	00110101	065	0x35	5	118	01110110	166	0x76	v
54	00110110	066	0x36	6	119	01110111	167	0x77	w
55	00110111	067	0x37	7	120	01111000	170	0x78	x
56	00111000	070	0x38	8	121	01111001	171	0x79	y
57	00111001	071	0x39	9	122	01111010	172	0x7A	z
58	00111010	072	0x3A	:	123	01111011	173	0x7B	{
59	00111011	073	0x3B	;	124	01111100	174	0x7C	|
60	00111100	074	0x3C	<	125	01111101	175	0x7D	}
61	00111101	075	0x3D	=	126	01111110	176	0x7E	~
62	00111110	076	0x3E	>	127	01111111	177	0x7F	DEL
63	00111111	077	0x3F	?					
64	01000000	100	0x40	@					

ملاحظـة: أول 32 رمـزاً الأولـى وآخـر رمـز هـي رمـوز تحكـم لا يمكـن طباعتها.

هذا الكتاب

- يهدف إلى تعريف الطلبة بالمبادئ الأساسية للغة C التي أصبحت شائعة الاستخدام على نطاق واسع وفي جميع مجالات العلوم التطبيقية والنظرية عن طريق العديد من البرامج المتنوعة التي تمتاز بالسهولة والبساطة.

- يضم أربعة عشر فصلاً ومجموعة من البرامج المتنوعة التي تمتاز بأسلوب سهل ومبسط وبذيل كل فصل العديد من التمرينات التي تجعل منه كتاباً منهجياً صالحاً للاستخدام ضمن المقررات الدراسية للطالب بالجامعات والمعاهد العليا.

- يبدأ بتعريف الأعداد والمتغيرات وكيفية استخدام دوال إدخال وإخراج البيانات مع شرح واف للتعبيرات الحسابية والمنطقية وجمل التخصيص والمؤثرات وجمل الاختيار والتبديل والتكرار والتفرعات.

- يشرح مؤثرات الازاحة والدوال والماكرو وكذلك المصفوفات العددية والحرفية وكيفية ارتباطها بالدوال والمؤشرات مع التركيز على هياكل البيانات وطريقة استخدام الملفات بأنواعها المختلفة.

الكتب الصادرة للمؤلف

